A REUSABLE SOFTWARE CATALOG INTERFACE

By

JOE E. SWANSON

Bachelor of Music Education

Central State University

Edmond, Oklahoma

1981

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1991

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	FINAL

4. TITLE AND SUBTITLE
A Reusable Software Catalog Interface

5. FUNDING NUMBERS

6. AUTHOR(S)
CPT Joe E. Swanson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Oklahoma State University
Stillwater, Oklahoma 74078

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Software reuse is the taking of any code or code segment and using it again to meet a specific need. Software reusability involves not only the reuse of software, but also how that software is designed. In other words, it involves designing totally self-contained software components. Each component should be easily modifiable to meet a potential user's need. This paper reports the prototypical implementation of a software library that uses the faceted cataloging scheme. It is designed to act as an interface between the user and a reusable software system. The prototype can be used to catalog and retrieve software components. Within the realm of this prototype, which is a domain-specific implementation, the number of facets used in the interface may be reduced while still achieving the same search and retrieval results.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Reuse, Reuse Catalog, Faceted Cataloging Scheme,	176
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASS	UNCLASS	UNCLASS	UL

A REUSABLE SOFTWARE CATALOG INTERFACE

Thesis Approved:

M. Samadzadeh
Thesis Adviser

D. E. Hendrix

D. C. Pandian

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Dr. Mansur H. Samadzadeh for his encouragement and advice throughout my thesis research. His willingness to give guidance and direction made this a meaningful learning experience. I also wish to thank Drs. G.E. Hedrick and John P. Chandler for serving on my graduate committee.

I also wish to thank Rohinton Mistry and Winai Wichaipanitch for the use of their Operating Systems II projects as test data.

A very special thanks to my wife, Terri, for her patience and understanding during the course work and thesis preparation; her assistance in preparing the manuscript was invaluable. To my sons, Jacob and James, thanks for not giving up when I would say "wait a minute." It's time to play ball.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Software reuse is currently a topic of great interest in the theory and practice of computer science in both academia and industrial research. Reducing software development cost through reuse of previously written and debugged code is practiced in many forms today and it "is an effective strategy for developers to reduce implementation cost" [PA90]; however, it is usually at the individual programmer level and very informal, based on one person's stock and his/her concept of what can be reused. By combining the efforts of a set of programmers at a given site, reuse libraries can be formed. Each programmer can then draw on source code from this library to accomplish a given task.

The next step is to make the use of the library efficient enough that the programmers would be willing to use it regularly, both as a customer and a contributor. The software components in the library must be readily accessible. A means must be devised to retrieve the closest match that meets a programmer's need with the least amount of modification.

In this thesis, the results of current work is examined and the most accurate library interface is identified. For this work, the interface is defined as the set of parameters that most clearly define a software component thus making its retrieval possible.

To demonstrate the feasibility of this task, a partial implementation of a software library is given. The data manipulated is the set of parameters used to describe each member of the software library (hereafter referred to as the repository). Identifying the most efficient method to store components is a topic for future research.

The concept of this implementation follows the basic structure proposed by Kazerooni-Zand, Samadzadeh, and George [KS90]. In their work, they deal with reuse at the object code level. Their system involves three major subsystems. The Identification Mechanism (IDM) is designed to select those components that meet a programmer's need. The Software Control Mechanism (SCM) is designed to provide access to different versions of a program or component; it receives input from the IDM. The Interface is designed to act as a pre and post compiler; it coordinates the micro-incremental compilation between their Reuse of Persistent Code and Object Code system and the compiler [KS90].

This thesis deals only with the action taking place within the IDM. The interface will be the parametrization of the detailed description of all components in the

software repository. In terms of the system defined by Kazerooni-Zand et al. [KS90], this is the Software Attribute Database. The level of reuse in this work is restricted to source code.

CHAPTER II

LITERATURE REVIEW

"Software reuse" is the taking of any code or code segment and using it again to meet a specific need. "Software reusability" involves not only the reuse of software, but also how that software is designed. In other words, it involves designing software components to be totally self-contained needing no outside code to accomplish their task, or if help is needed, knowing where to find it. Each component should be easily modifiable to meet a potential user's need.

As stated by Biggerstaff and Perlis, "software reuse is the reapplication of a variety of kinds of knowledge about one system to another similar system in order to reduce the effort of development and maintenance of that other system" [BP89a]. Reusability can mean the reuse of ideas as well as the reuse of components. So when an algorithm is used repeatedly, the algorithm is being reused.

According to Biggerstaff and Perlis [BP89a], software reuse has been in use for a number of years to a limited and informal extent. In many cases, it is limited to the knowledge of an earlier system that is similar to the

current project, or modules from other projects or systems that lend themselves to accomplishing a specific task. Reuse figures have been reported as high as 80%, but on average the figure is considered to be much lower, possibly as low as 5%. Other items of reuse such as life-cycle objects (requirements, designs, and test plans) have even lower rates of reuse [FG89] [FG90].

Portability can be considered a subarea of reusability. Portability is defined in general terms as "running a program in a host environment that is somehow different from the one for which it was designed" [JA88].

Portability became popular with the advent of the C programming language though it was an issue long before the appearance of C. As early as the late 1950's and early 1960's, COBOL was defined as a standard business language. Two corporations developed compilers late in 1960 and exchanged code [JA88]. With a minimum number of modifications, the code ran on both machines. However, just because a piece of code is written in COBOL or C (i.e., a "portable" language), it does not mean the program is necessarily portable [JA88]. Likewise, because a portion of a program is reused, it does not mean that the code is reusable in the sense of software reusability.

Basili approaches software reuse from the viewpoint of software maintenance [BA90]. He details three models that can be used during the process of updating an old system

while using the old code as a bank of reusable code to create the new system. These three models are described below.

The "quick-fix" model uses the existing system. Modifications are made to the source code and documentation as necessary.

The "iterative-enhancement" model is designed to be used when all the requirements of the system are not known or the developer is not capable of building the full system. This model, which is well-suited to software maintenance, also starts with the existing system together with its requirements and documentation. It is an evolutionary process that allows for updating and redesigning of the system based on analysis of the system as the work progresses.

The "full-reuse" model begins with the requirements of the new system and uses components of the old system as needed along with other components that may be in the software repository. This model assumes that the existing components are well documented or that they are documented as they are added to the repository [BA90].

Basili does not go into the detail of cataloging the components for his models. However, in the case of the full reuse model, a catalog interface describing each member in the set of components to be used for the reuse project could enhance the model's usefulness.

Different Approaches to Reusability

Various concepts of software reusability have been presented by researchers. Prieto-Diaz and Freeman present a software classification scheme used to catalog pieces of software for future use [PF87]. The scheme enables the user to give parameters for a search and then the system selects and recommends software modules that match or closely match the parameters. The parameters are part of the interface used for the presented system.

Prieto-Diaz and Freeman's scheme is divided into two major areas: functionality and environment. Each area is further subdivided into three parameters. Functionality is described by function, objects, and medium. Function is self-explanatory; it describes what the software does, i.e., the action. Object describes the objects manipulated by the function, and medium describes the data structure(s) or device(s) that the function uses.

The environment is broken down into system type, functional area, and setting. The system type "refers to functionally identifiable, application-independent modules, usually including more than one component." Functional area describes "application-dependent activities usually defined by an established set of procedures." And the setting describes "where the application is exercised" [PF87].

The facets of this scheme form a sextuple that describes the respective components in the software reposi-

tory. To insure a uniform meaning for the sextuple, vocabulary control is imposed to facilitate comparisons. A thesaurus is used to convert all of the definitions to descriptive words of like meaning. This lends consistency to the comparisons.

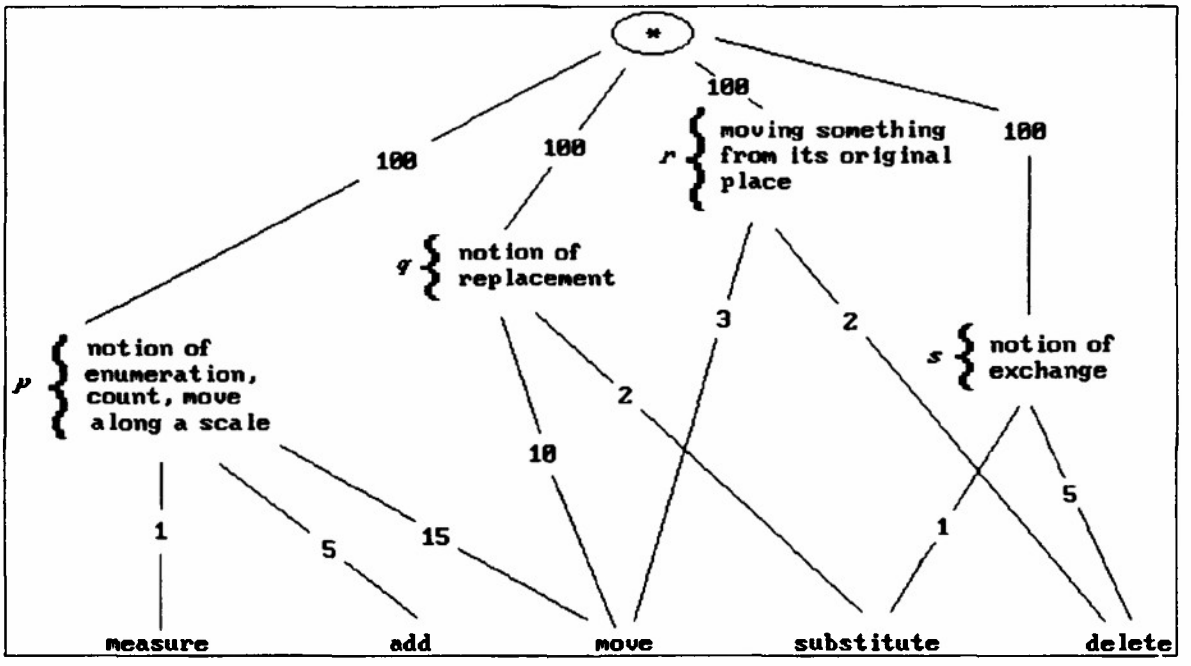


Figure 1. A Partial Weighted Conceptual Graph for the Function Facet.

A partially weighted graph (Figure 1) is used to help identify a closely related term when one of the members of the sextuple is not found. The graph is a DAG, Directed Acyclic Graph, with each of the nodes being "supertypes that denote general concepts relating two or more terms" [PF87]. Weights are assigned to the edges using a software

engineering perspective; the closer the perceived proximity of the terms, the lower the weight assigned to the edge connecting the two terms. When a query is made and a term is not matched, the graph is consulted to find closely matching terms, and this gives the user a set of closely related components to choose from.

To define the software components further, Kazerooni-Zand, Samadzadeh, and George start with Prieto-Diaz and Freeman's faceted scheme and add the implementation environment. It is composed of two elements: language and machine [KS90].

The Reuse of Persistent Code and Object Code (ROPCO), the system proposed by Kazerooni-Zand, Samadzadeh, and George, deals with reuse at a very low level. Machine dependency becomes much more important when attempting the reuse of object code as compared with source code.

The ROPCO system is composed of three major subsystems: the Identification Mechanism (IDM), the Software Control Mechanism (SCM), and the Interface. The function of the IDM is to identify and select the programs or modules that meet the user or programmer's requirements. Each element in the system has a record in the Software Attribute Database (SADB). This record is a unique identifier to its assigned element and is used throughout the ROPCO system to identify a particular element. The IDM prompts the user for the functional and environment attributes as defined by Prieto-

Diaz and Freeman [PF87], and the implementation environment as defined by Kazerooni-Zand et al. [KS90]. Using the attributes input by the user, the SADB is accessed and one or more elements are chosen; the selections are based on the descriptions in the SADB and the proximity distance model [PF87].

Kernighan uses the UNIX[1] system as an example of an environment that facilitates reusability. By using system utilities, he proposes that one can build more complicated programs and utilities using shell scripts or the source code of the system utilities [KE84].

Caldiera and Basili approach software reusability by splitting the traditional life cycle models into two parts. The first part is the project and the second is the factory [CB91]. The project delivers the software system to the customer. As a need for a component is developed by the project, it is sent to the factory. The factory deals with extracting and packaging reusable components. It also has to have a detailed knowledge of the project it is supporting.

In their system, all components, which are selected for possible addition to the software repository, go through a two-phase evaluation. Phase one, the identification phase, consists of three steps: the definition of the reusability

---

[1]UNIX is a trademark of AT&T.

attributes model, extraction of the component, and application of the model. Phase two, the component qualification phase, is composed of six steps: generation of the functional specification; generation of the test cases; classification of the component; development of the reuser's manual; storage; and feedback.

Phase one is automated using software models and metrics. Components that pass this phase are analyzed by a domain expert. Any component whose functional specification is not relevant, or is incorrect, is thrown away. The reason for eliminating the component is documented; this aids in the development of future reusable components.

A component is then run through a series of tests that are based on its functional specification. If the tests are failed, then once again the component is discarded and the reason is documented. At this point, any component that has survived is classified and documented for reuse. Each component is made an autonomous unit capable of being compiled without the addition of other files; any information that may have resided in other files (C include files for example) would have to be included in the component [CB91].

Purtilo and Atlee [PA90] introduced a language called NIMBLE that is used to ease the introduction of reusable modules into new applications. With NIMBLE, the difference between parameter orders is removed. NIMBLE provides

parameter coercion capabilities without changing the source code of a module. The actual and formal parameter lists are referred to as interface patterns. A mapping from the actual parameter list to a new "parametric" takes place to meet the needs of the module being called [PA90].

One of the motivations for the design of NIMBLE was the likelihood that modules that are semantically identical, but may be structurally different, are likely to become more prevalent as reusable software systems are developed. NIMBLE can be used to bridge the module interface gap and also it can widen the domain to which a module can be applied [PA90].

Frakes and Gandel [FG89] [FG90] discuss various methods of representing reusable software components including library science, knowledge based methods, and hypertext techniques. They define representation "as a language (textual, graphical, or other) used to describe a set of objects" [FG89].

The faceted classification scheme of Prieto-Diaz and Freeman [PF87] falls under the category of library science. To compensate for the narrow focus of an enumerated system such as the Dewey Decimal System, faceted schemes allow a subject area to be broken down into fundamental parts. These parts can then be synthesized to develop more descriptive representations [FG89].

Commercial component libraries such as GRACE (Generic Reusable ADA Components for Engineering) do exist. For instance, GRACE allows a user to choose from a list of ADA packages to accomplish common tasks such as managing stacks and queues. It uses a knowledge engineering approach to represent the software modules [FG89].

Chatterbox[2], a graphical user interface library available from Courseware Applications, Inc., is a library of graphic routines for the C language. It performs the dirty work to create menu bars with pull-down menus and dialog boxes with keyboard and mouse support using the graphics routines of the supported compilers. The user decides what (s)he wants and then includes the necessary routines in his/her code. The routines are described in the Chatterbox reference manual. There is not an automated method available to pull routines together; the user must decide what (s)he wants by looking at the manual or by using past experience with the library. The user then goes to the Chatterbox source code and selects the necessary module or modules [SC90].

Reusability in Practice

Prieto-Diaz [PR91] states that a classification scheme for reusable software must meet the following criteria:

---

[2]Chatterbox is a trademark of Courseware Applications, Inc.

1. It must accommodate continually expanding collections, a characteristic of most software organizations,
2. It must support finding components that are similar, not just exact matches,
3. It must support finding functionally equivalent components across domains,
4. It must be very precise and have high descriptive power (both are necessary conditions for classifying and cataloging software),
5. It must be easy to maintain, that is, add, delete, and update the class structure and vocabulary without any need to reclassify,
6. It must be easily usable by both the librarian and end user, and
7. It must be amenable to automation.

Prieto-Diaz [PR91] has worked with large corporations such as GTE Data Services (GTE DS) and the CONTEL Technology Center. While working at GTE DS, a 14 percent reuse factor with a savings of 1.5 million dollars reportedly has been realized the first year. A library "asset" was defined as any facility that could be reused in the production of software with the initial emphasis being reusable software components. It was found that 38 percent of the 190 assets in the library the first year were being "actively reused." The majority of the items in the library were components

with greater than 10,000 lines of code; the larger components provided a greater savings when reused. The faceted scheme was more effective in "domain specific collections than for broad, heterogeneous collections."

In his work at CONTEL, domain-specific reusable software repositories were established. The library system at CONTEL was developed to enable its easy integration into existing environments. This is an example of a library system that would be capable of being instantiated into multiple environments.

To encourage the use of the repository, monetary incentives were used to motivate the programmers, referred to as authors or asset providers, to write their software with reusability in mind, and to submit the new material for inclusion in the repository.

CHAPTER III

PROTOTYPE IMPLEMENTATION

General Overview

The main focus of this thesis is a prototype implementation of a software library. The prototype can be used to: (a) catalog software components using the faceted cataloging scheme presented by Prieto-Diaz and Freeman [PF87], and (b) retrieve components from a software repository. This implementation is used to catalog components at the source code level (other possibilities include the specification, design, and object code levels).

The system has three major sub-sections: adding components, querying the system for candidate components, and the common vocabulary or thesaurus. This system works within the confines of the Identification Mechanism (IDM) as defined by Kazerooni-Zand et al. [KS90].

Concept of the Design

The prototype was implemented using the C programming

language on an IBM PC/AT[3] compatible. Microsoft Quick C compiler Version 2.5 was used for all coding and compiling.

The prototype requires eight supporting data files. The common vocabulary (hereafter referred to as the thesaurus) is contained in seven data files with the eighth file being the Software Attribute Database (SADB) as defined by Kazerooni-Zand et al. [KS90]. Each facet has its own thesaurus data file. The facets used for this prototype and the associated file names of their respective thesauruses are shown in TABLE I. The thesaurus file for language is included for information only; language is not used as a parameter during queries.

TABLE I

FACETS WITH FILE NAMES

function	funct.ths
object	object.ths
medium	medium.ths
system type	systype.ths
functional area	funcarea.ths
setting	setting.ths
language	lang.ths

Each line of a file is a comma-delimited list of descriptors with the first word on the line being the

[3]IBM PC/AT is a trademark of International Business Machines Corp.

primary descriptor of that line. TABLE II shows a subset of the "function" common vocabulary. The first word of each line is the primary descriptor that will be used to define a component for this facet for all components described by one of the descriptors on that line. Given the set of descriptors "add,increment,total,sum" from the "function" thesaurus file, the user could enter any of the words as a legal input. If a user enters the word "sum," the thesaurus searches the data file of the respective facet. When sum is found, "add" is substituted for the user's input.

TABLE II

SAMPLE THESAURUS ENTRIES

add,increment,total,sum
close,release,detach,disconnect,free
compare,test,relate,match,check,verify
complement,negate,invert
measure,count,advance,size,enumerate,list

The thesaurus is accessed every time a user enters a descriptor (i.e., during cataloging and making queries). If the user enters a word not included in the thesaurus, (s)he is presented with a list from which to choose.

All of the supporting database functions are provided by source code reused from an earlier academic project (16 percent of the total lines of code are reused). Each of the

data files are indexed and accessed using inverted lists. With each data file, there are three supporting files. TABLE III shows the data files with their respective supporting files and TABLE IV gives the purpose of each of the data files.

TABLE III

NAMES OF THE DATA FILES
WITH SUPPORTING
FILE NAMES

Data File	Word File	Inverted List	Inverted List Index
ru.dta	ru.wrd	ru.inv	ru.vdx
funct.ths	funct.wrd	funct.inv	funct.vdx
object.ths	object.wrd	object.inv	object.vdx
medium.ths	medium.wrd	medium.inv	medium.vdx
systype.ths	systype.wrd	systype.inv	systype.vdx
funcarea.ths	funcarea.wrd	funcarea.inv	funcarea.vdx
setting.ths	setting.wrd	setting.inv	setting.vdx
lang.ths	lang.wrd	lang.inv	lang.vdx

TABLE IV

FILE EXTENSION DEFINITIONS

File Extension	Definition
.wrd	contains all words in the data file with their line numbers that are used to create the inverted list
.inv	an inverted list for the data file
.vdx	the index file for the inverted list

The thesaurus was developed with the intended use of cataloging Operating Systems II class projects and their components as entries. This is a domain-specific implementation as was the implementation used at CONTEL by Prieto-Diaz [PR91]. It is possible to change the domain orientation by changing the common vocabulary in the thesaurus data files.

For this prototype implementation, the vocabulary consists of those terms needed to define the set of software components used as sample input. The list was limited to those terms needed to give an adequate sampling of the programs usefulness. Developing a comprehensive vocabulary for the operating systems domain is beyond the scope of this thesis.

User Interface

The prototype has three major sub-sections: adding components, querying the system for candidate components, and the common vocabulary or thesaurus.

The system is menu driven with the video display partitioned into multiple text windows. With few exceptions, pressing one key is all that is required to maneuver through the program. Figure 2 shows the opening menu where the user is presented with three choices. Each of the prompts are self-explanatory.

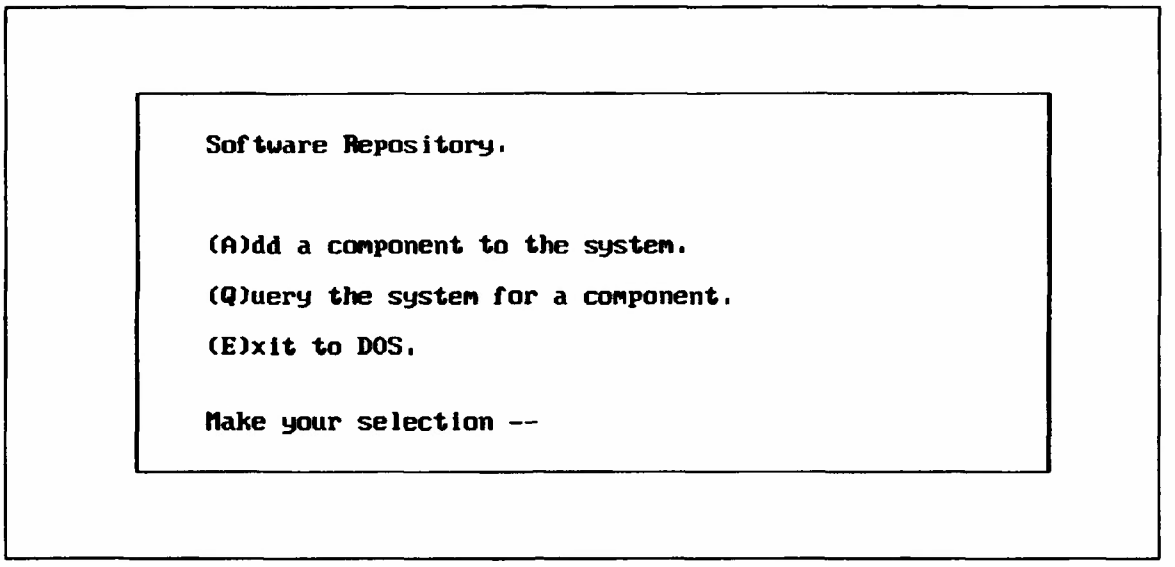


Figure 2. Software Library Program Opening Menu.

After electing to add a component to the repository, the user is prompted for the component name; the system must be able to find the component before it can be added to the repository. When the system is ready for the user to assign attributes to the module, the user will see the screen shown in Figure 3.

Attributes are entered as prompted. After all the descriptors have been input, the user is given a chance to make changes. When the descriptors are accepted, the component will be added to the system. No maintenance is required for this phase of the system.

When the user chooses Q at the opening menu, the system updates the Software Attribute Database (SADB) to insure that additions are included. The entering of attributes for

a query is done exactly as assigning attributes to a new component.

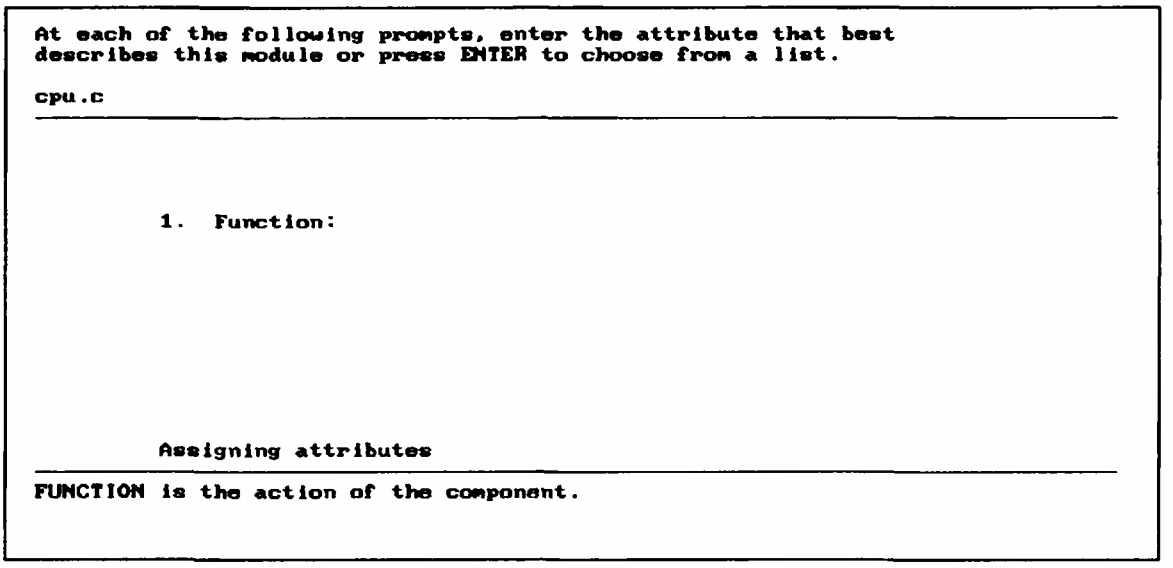


Figure 3. Data Entry Screen.

When the list of desired attributes has been accepted, SADB is searched for all components that meet one or more of the desired attributes. After all exact matches are displayed, the user is presented with a message at the bottom of the screen containing a component name and the percentage of the attributes matched. The user can make the decision to view or bypass a component.

After all the possible candidates have been viewed or passed by, the user is asked if (s)he wants to keep the list of candidates. If a negative response is given, the list is discarded and the program returns to the main menu. If

accepted, the user is prompted for a file name and up to two lines of comments. Then the candidates with their attributes are dumped into the given file. While viewing the result of a system query, the user can press D and the system will display the facet definition at the bottom of the screen.

During the assigning of attributes and while making queries of the system, the thesaurus is available by pressing the ENTER key. When the key is pressed, a list of possible choices for the current facet is displayed on the right side of the screen. Also, if the user enters a descriptor not included in the common vocabulary, a list will appear on the right side of the screen. By selecting a letter corresponding to one of the given choices, the program inserts the selected descriptor. N and P are pressed to get the next and previous lists (see Figure 4).

The user cannot make any changes to the software repository from inside the program except for the addition of modules. To delete components from SADB, any ASCII text editor can be used. Each new line in the data file named ru.dta is a set of component descriptors. The line is a comma-delimited list containing the component name, function, object, medium, system type, functional area, setting, and language in that order. A ninth entry may be present if the user has assigned a custom attribute from a user-supplied thesaurus file. To delete a component from

SADB, the user must find the line containing its name and delete it. The next time the system is used to make a query, the SADB supporting files will be updated automatically.

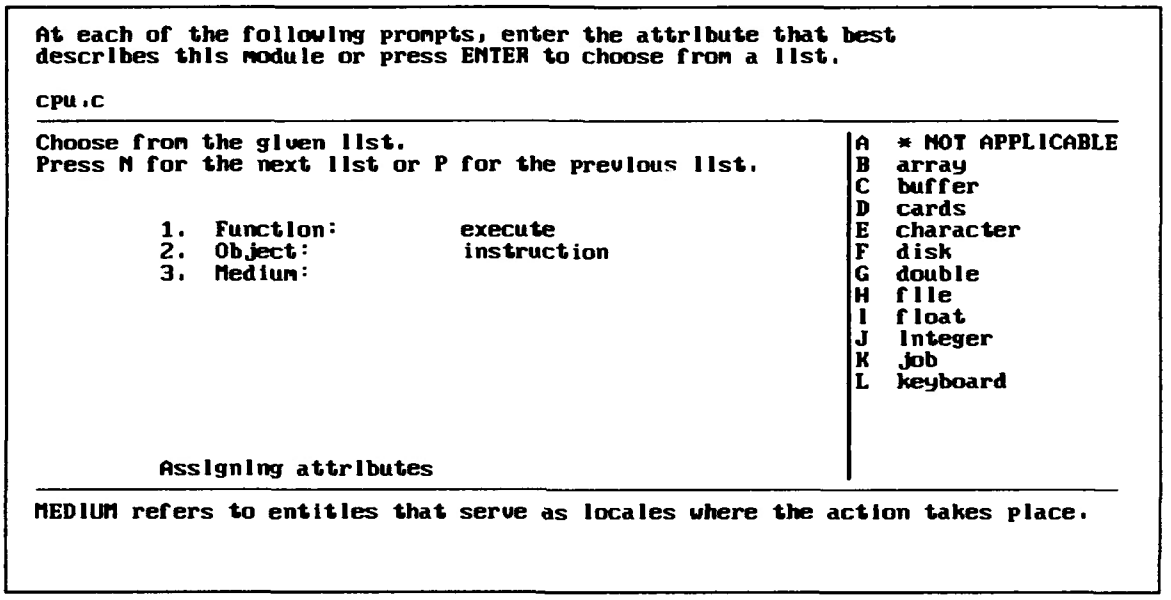


Figure 4. Data Entry Screen with Thesaurus.

To update one of the common vocabulary lists, any ASCII text editor can be used. The thesaurus entries are stored in a comma-delimited list with the first word of the list being the primary key for that particular set of descriptors. Each line of descriptors must be terminated with a carriage return (new line character, "\n"). The user can add or delete lines of descriptors. Also the user can add to or delete from the existing words. After changes are

made, at least one of the supporting files for the particular facet must be deleted. The next time the thesaurus accesses the list for the updated facet, its supporting files will be updated.

Sample Queries

The test data for this prototype implementation consists of components of varying lengths and purposes. Programs, of three programmers from a graduate level operating systems course, were broken into code segments of varying sizes based on each programmer's comments. A total of forty-three components were cataloged using the prototype.

Components range in size from three lines (simple string manipulations) to more than 700 lines. Some of the larger modules consist of multiple C functions that work in concert to accomplish an assigned task; in other cases, a module consists of a single function (some containing more that 250 lines of code). As expected, it was found that the smaller, more specialized the component, the easier it was to classify.

Using the program specification [SA91] that was used to develop the programs in the data set, a set of queries were developed to find components to meet the requirements of the principal routines: loader, memory, cpu, spooler, and scheduler (shown in order in TABLE V). The queries were

performed using four and six facets. Because this is a domain-specific implementation, the system type and setting facets were omitted from one set of queries. This was done to determine if the removal of the aforementioned facets changed the results of the queries measurably. There is a small set of components of a generic nature that can be used in any setting or system type as needed; however, in this sampling, their presence is insignificant.

TABLE V

SAMPLE SYSTEM QUERIES

Function	Object	Medium	System Type	Funct. Area	Setting
load	job		operating system	job io	academic
access	memory	array	operating system	memory management	academic
execute	instruction	integer	operating system	processing	academic
reads	card stack		operating system	job io	academic
schedule	jobs		operating system	scheduling	academic

As can be seen in TABLE V, the medium column is not used in every query. If a dominant medium could not be identified in a component during the cataloging of the component or in a component's specification, as in the case of the queries in TABLE V, it was omitted.

Query Results

To accomplish a partial validation of this prototype implementation, ten sample queries were completed; five queries using four facets (function, object, medium, functional area), and five queries using six facets (function, object, medium, system type, functional area, setting). The language facet was not used as a descriptor for queries because all of the components in the system are written in the same language, so this facet would not provide any useful information during a query.

Because the prototype finds every component in the system that meets one or more facets, the queries using six facets list some candidate components that meet only the system type and setting, or both. In this domain-specific implementation with the given data set, these facets do not provide any useful information. TABLE VI gives a summary of the number of components found using both four and six facets.

The left column of TABLE VI shows the number of attributes matched (i.e., if six attributes are entered as a query, "Exact" indicates an exact match of all the requested attributes; -1 indicates that five attributes are matched; -2 indicates that four attributes are matched; etc.). Columns s_1 through s_5 give the number of components found in samples 1 through 5 for the given number of facets.

TABLE VII shows at what point a search should be considered successful. As previously stated, the prototype delivers all components that meet one or more of the requested attributes. At some point, the components returned are of little or no value because of the particular facets matched. Because the results are based solely on attributes matched, an additional or alternate method must be developed to discriminate between the particular facets matched. This is a topic for future study. But for the purposes of this prototype, any component that is returned that contains two or more mismatched attributes should be discarded. The components represented in TABLE VII above the double line should be considered valid candidates, those below, invalid.

TABLE VI

SUMMARY OF QUERIES

	Four Facets					Six Facets				
	s_1	s_2	s_3	s_4	s_5	s_1	s_2	s_3	s_4	s_5
Exact	2	3	–	2	2	2	3	–	2	2
-1	2	–	3	–	–	2	–	3	–	–
-2	3	–	2	–	3	4	–	2	1	3
-3	–	3	4	–	–	14	3	2	15	16
-4	–	–	–	–	–	–	13	13	–	–
-5	–	–	–	–	–	–	–	1	–	–
Total	7	6	9	2	5	22	19	21	18	21

TABLE VII

SUMMARY OF VALID AND INVALID QUERIES

	Four Facets					Six Facets				
	s_1	s_2	s_3	s_4	s_5	s_1	s_2	s_3	s_4	s_5
Exact	2	3	–	2	2	2	3	–	2	2
-1	2	–	3	–	–	2	–	3	–	–
-2	3	–	2	–	3	4	–	2	1	3
-3	–	3	4	–	–	14	3	2	15	16
-4	–	–	–	–	–	–	13	13	–	–
-5	–	–	–	–	–	–	–	1	–	–
Total	7	6	9	2	5	22	19	21	18	21

CHAPTER IV

SUMMARY AND FUTURE WORK

The aim of this thesis was to identify an accurate software catalog interface capable of correctly identifyinc software components and hence enabling their retrieval. The vehicle used to accomplish this was a partial, prototypical implementation of a software library.

Using the faceted cataloging scheme of Prieto-Diaz and Freeman [PF87] in a domain-specific library, it was decided that two facets (system type and setting) should be eliminated from the query process. This is due to the narrowness of the focus of the domain. As was reported in Chapter III, within the sample domain of this thesis, the results were the same with or without the use of the system type and setting for valid queries.

The interface developed as part of this thesis can be instantiated into different domains by changing the vocabulary. Also, it can be used at different levels of reuse other than source code such as design, specification, or object code levels.

In order to validate the prototype fully, it must be installed in an industrial setting to verify its performance

over time. A system librarian would enhance the usefulness of the library by standardizing component descriptions. The program itself is easy to use, but the task of defining the software components is a time-consuming effort; time that a typical programmer may not care to expend (invest) to accomplish the task correctly. Programmers would have to be encouraged to develop their components with reusability in mind as was done at GTE DS and CONTEL [PR91].

To enhance the usefulness of this prototype implementation, the maintenance functions need to be automated. Also, the system needs a method to refine the results of the queries. The user should be given an indicator of the complexity required to modify the component to meet his/her need.

Future study is needed to determine the method used to store components ("assets" as defined by Prieto-Diaz [PR91]) most efficiently. Data compression may be one method to integrate into this prototype in its current form.

For the concept of reusability to become a widespread reality, it will require dedicated software librarians to accomplish the task of managing the repositories that will grow with time. This may open other specialized areas within the field of computer science.

BIBLIOGRAPHY

[BA90] Basili, Victor R., "Viewing Maintenance as Reuse-Oriented Software Development," *IEEE Software*, January 1990, pp. 19-25.

[BP89a] Biggerstaff, Ted and Alan J. Perlis, *Software Reusability: Concepts and Models*, Addison-Wesley Publishing Co., Vol. 1, 1989.

[BP89b] Biggerstaff, Ted and Alan J. Perlis, *Software Reusability: Applications and Experience*, Addison-Wesley Publishing Co., Vol. 2, 1989.

[BP90] Bollinger, T.B. and S.L. Pfleeger, "Economics of Reuse: Issues and Alternatives," *Information and Software Technology*, December 1990, pp. 643-652.

[BR87] Biggerstaff, Ted and Charles Richter, "Reusability: Framework, Assessment, and Directions," *IEEE Software*, March 1987, pp. 41-49.

[CB91] Caldiera, Gianluigi and Victor Basili, "Identifying and Qualifying Reusable Software Components," *IEEE Computer*, February 1991, pp. 61-70.

[CH84] Cheatham, Jr., T.E., "Reusability Through Program Transformations," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 5, September 1984, pp. 589-594.

[CL84] Cheng, Thomas T., Evan D. Lock, and Noah S. Prywes, "Use of Very High Level Languages and Program Generation by Management Professionals," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 5, September 1984, pp. 552-563.

[CO90] Cox, Brad J., "Planning the Software Industrial Revolution," *IEEE Software*, November 1990, pp. 25-33.

[CU85] Curran, Anne Marie, *On Design and Implementation of an Environment for Reusable Software*, Ph.D. Dissertation, University of Southern California, May 1985.

[FG89] Frakes, W.B. and P.B. Gandel, "Representation Methods for Software Reuse," *Proceedings of Ada Technology in Context: Application, Development, and Deployment*, Pittsburgh, Pennsylvania, October 1989, pp. 302-314.

[FG90] Frakes, W.B. and P.B. Gandel, "Representing Reusable Software," *Information and Software Technology*, December 1990, pp. 653-664.

[GO86] Goguen, Joseph A., "Reusing and Interconnecting Software Components," *IEEE Computer*, Vol. 19, February 1986, pp. 16-28.

[JA88] Jaeschke, Rex, *Portability and the C Language*, Hayden Books, Indianapolis, Indiana, 1988.

[KG87] Kaiser, Gail E. and David Garlan, "Melding Software Systems from Reusable Building Blocks," *IEEE Software*, July 1987, pp. 17-24.

[KE84] Kernighan, Brian W., "The UNIX System and Software Reusability," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 5, September 1984, pp. 513-518.

[KS90] Kazerooni-Zand, M., M. H. Samadzadeh, and K.M. George, "ROPCO: An Environment for Micro-Incremental Reuse," *Proceedings of the IEEE International Phoenix Conference on Computers and Communications*, Scottsdale, Arizona, March 1990, pp. 347-354.

[MA84] Matsumoto, Yoshihiro, "Some Experiences in Promoting Reusable Software: Presentation in Higher Abstract Levels," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 5, September 1984, pp. 502-513.

[PA90] Purtilo, James M. and Joanne M. Atlee, "Improving Module Reuse by Interface Adaptation," *Proceedings of the International Conference on Computer Languages*, New Orleans, Louisiana, March 1990, pp. 208-217.

[PF87] Prieto-Diaz, Rubén and Peter Freeman, "Classifying Software for Reusability," *IEEE Software*, January 1987, pp. 6-16.

[PR91] Prieto-Diaz, Rubén, "Implementing Faceted Classification for Software Reuse," *Communications of the ACM*, May 1990, pp. 88-97.

[SA91] Samadzadeh, Mansur H., "Operating Systems II Course Project Specification," Oklahoma State University, Stillwater, Oklahoma, Spring 1991.

[SC90] Schaefges, Thomas M., *Chatterbox Reference Manual*, Courseware Applications, Inc., Champaign, Illinois, 1990.

[SO89] Sommerville, Ian, *Software Engineering*, Addison-Wesley Publishing Co., Third Edition, 1989.

[ST84] Standish, Thomas, "An Essay on Software Reuse," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 5, September 1984, pp. 494-497.

APPENDIXES

APPENDIX A

SOFTWARE REPOSITORY USER'S GUIDE

SOFTWARE REPOSITORY

USER'S GUIDE

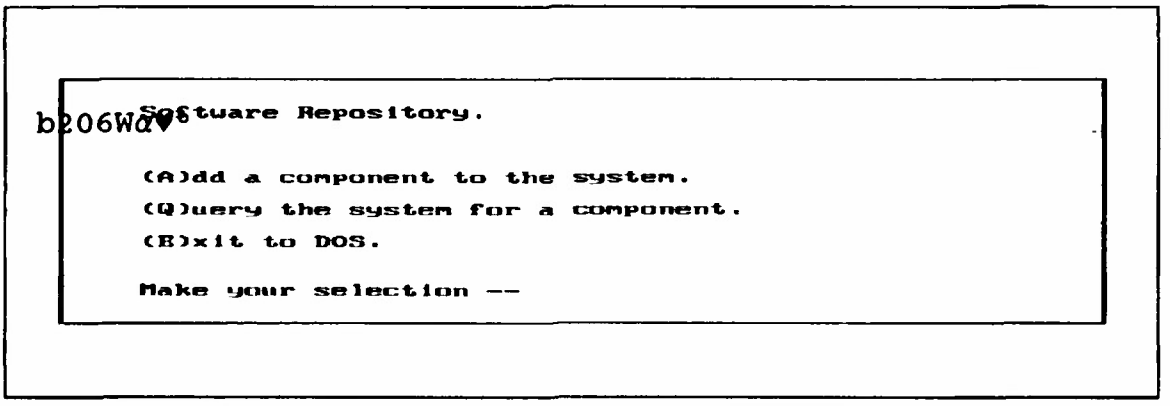
1. General Overview

This program is used to catalog software components using the faceted cataloging scheme presented by Rubén Prieto-Diaz and Peter Freeman [PF87], and to retrieve components from the software repository. This implementation is aimed at cataloging components at the source code level. The system has three major sub-sections: adding components, querying the system for candidate components, and the common vocabulary or thesaurus.

2. Adding Components

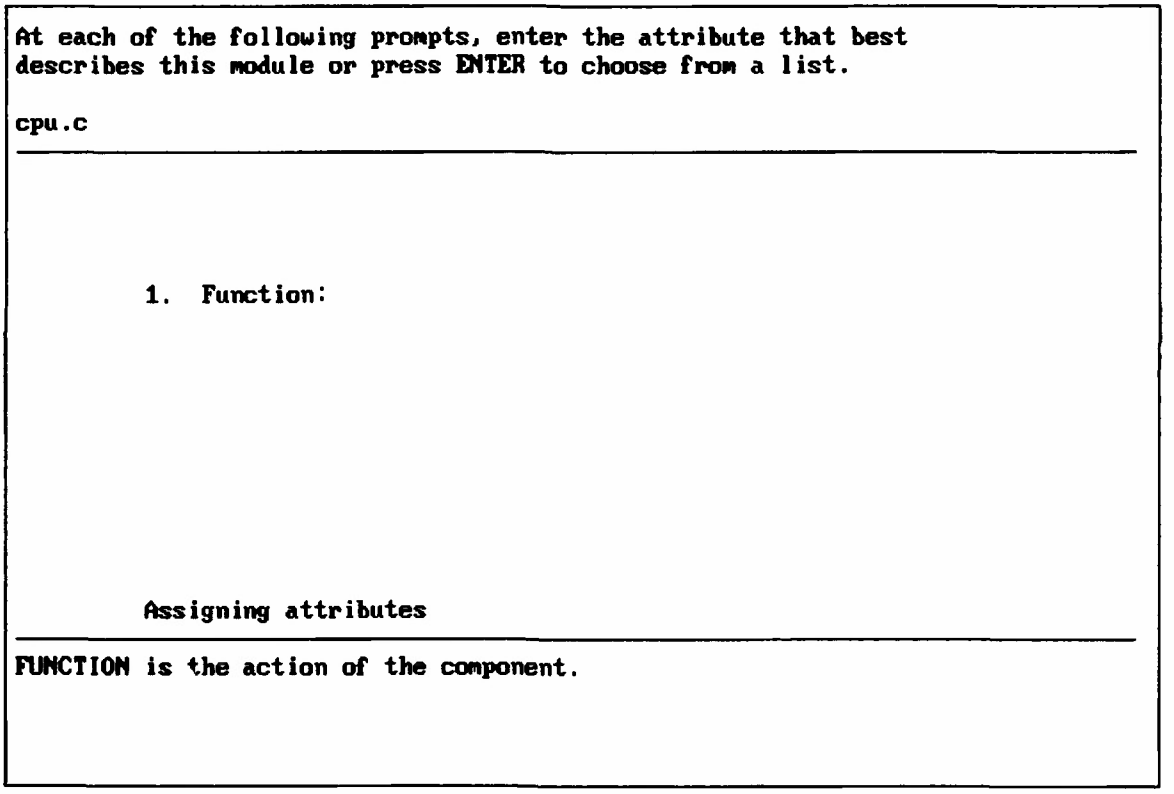
At the opening menu, you will be presented with three choices as follows.

The Opening Menu



Choose A to add a component to the repository. The prompts are self-explanatory. Next you will be prompted for the component name; the system must be able to find the component before allowing it to be added to the repository. When the system is ready for you to assign attributes to the module, you will see the following screen.

Attribute Assignment Screen



Enter the attributes as prompted. After all the descriptors have been input, you can make any changes. When you accept the entered list of descriptors, your component will be added to the system.

3. Making Queries of the System

From the opening menu choose Q. The system will then update the Software Attribute Database (SADB) to insure that the additions can be found. When making a query, enter the desired attributes just as you did when adding a component to the repository.

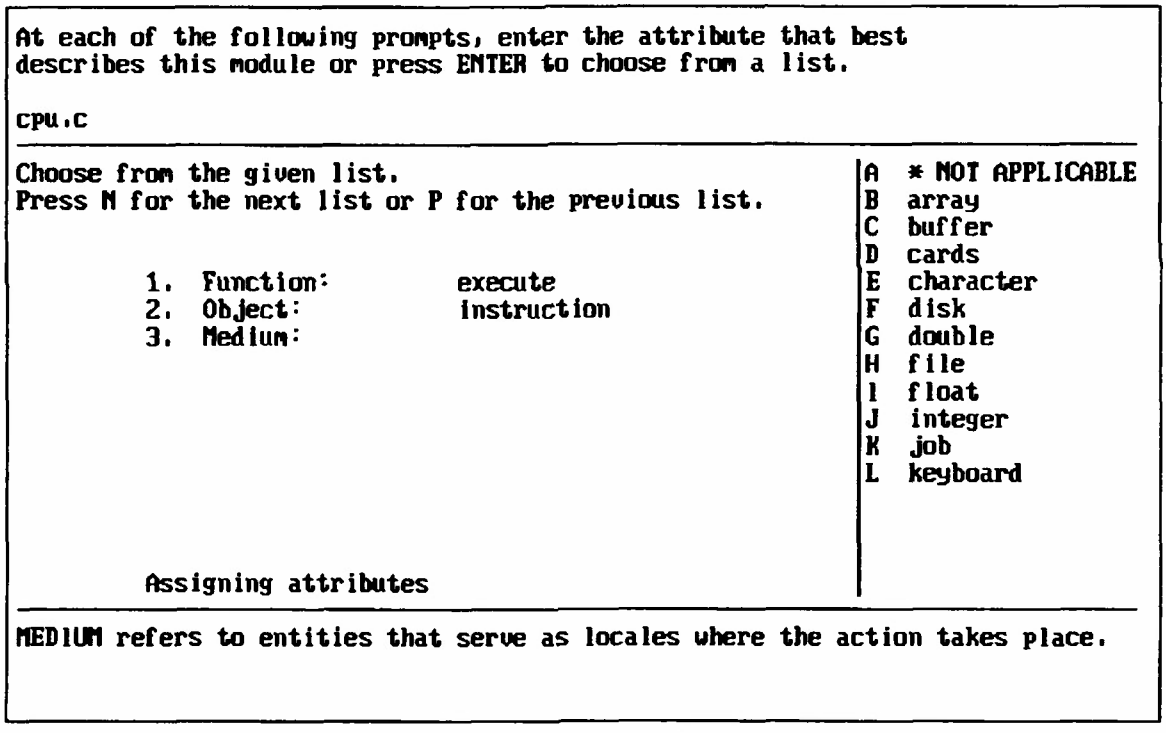
When you have accepted the list of desired attributes, SADB is searched for all components that meet one or more of the desired attributes. After all exact matches are displayed, a message will appear at the bottom of the screen containing a component name and the percentage of the attributes matched. If you wish to view the components attributes, press Y. After all the possible candidates have been view or passed by, you will be asked whether you want to keep the list of candidates. If you decline, the list is discarded and you are returned to the main menu. If you accept, you are prompted for a file name and up to two lines of comments. Subsequently, the candidates with their attributes are dumped into the file you have entered. While viewing the result of a system query, you can press D and the system will display the facet definition at the bottom of the screen.

4. Using the Thesaurus

During the assigning of attributes and while making queries of the system, the thesaurus is available by pressing the ENTER key. When the key is pressed, a list of

possible choices for the current facet is displayed on the right side of the screen. Also, If you enter a descriptor not included in the common vocabulary, a list will appear on the right side of the screen. Select the letter corresponding to your choice and the program will insert it for you. You may press N or P to get the next or previous lists.

Facet Options for the Attributes



5. Miscellaneous

You cannot make any changes to the software repository from inside the program except for the addition of modules. To update the common vocabulary or to delete components from

the system, see the System Administrator's Guide (Appendix B).

You may develop your own additional facet, if you so desire, to further define your software component(s). To do this, you need an ASCII text editor. Give the file whatever name you desire. In the file, arrange the descriptors so that each new line contains a new set of descriptors as shown below.

SAMPLE THESAURUS ENTRIES

```
add,increment,total,sum
close,release,detach,disconnect,free
compare,test,relate,match,check,verify
complement,negate,invert
measure,count,advance,size,enumerate,list
```

Each line must be terminated with a carriage return. As the C programming language is case sensitive, all searches of the descriptors are done in lowercase. Only the first word of the line can contain both upper and lowercase letters. Make sure the first word is the way you want is displayed. All remaining letters and words on the line should be in lower case to facilitate the search routines.

During the assigning of attributes or while making a query, you will be presented with an eighth prompt: "Other." At this prompt enter the user-defined facet. The

program will then prompt you for the file name of the file containing the common vocabulary for this facet. It is also possible to assign a second attribute from one of the program thesaurus files by entering the proper file name at the prompt.

APPENDIX B

SOFTWARE REPOSITORY SYSTEM ADMINISTRATOR'S GUIDE

SOFTWARE REPOSITORY

SYSTEM ADMINISTRATOR'S GUIDE

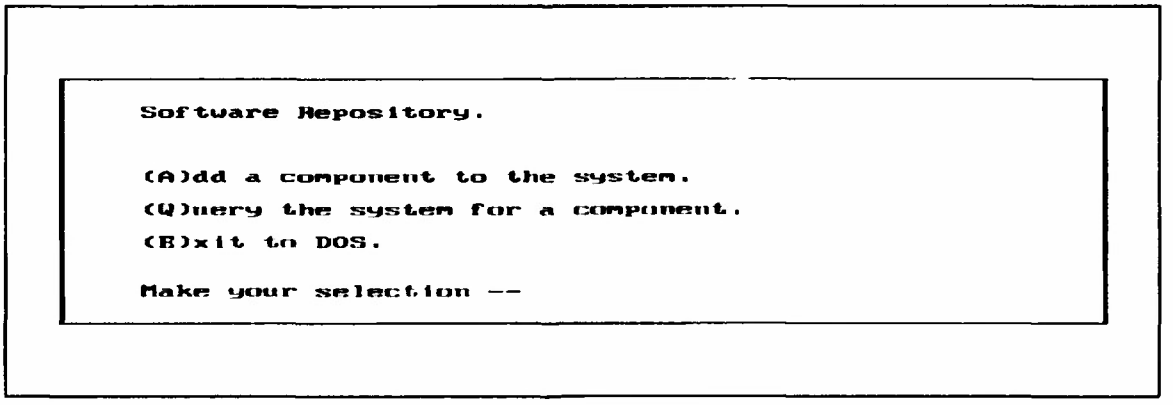
1. General Overview

This program is used to catalog software components using the faceted cataloging scheme presented by Rubén Prieto-Diaz and Peter Freeman [PF87], and to retrieve components from the software repository. This implementation is aimed at cataloging components at the source code level. The system has three major sub-sections: adding components, querying the system for candidate components, and the common vocabulary or thesaurus.

2. Adding Components

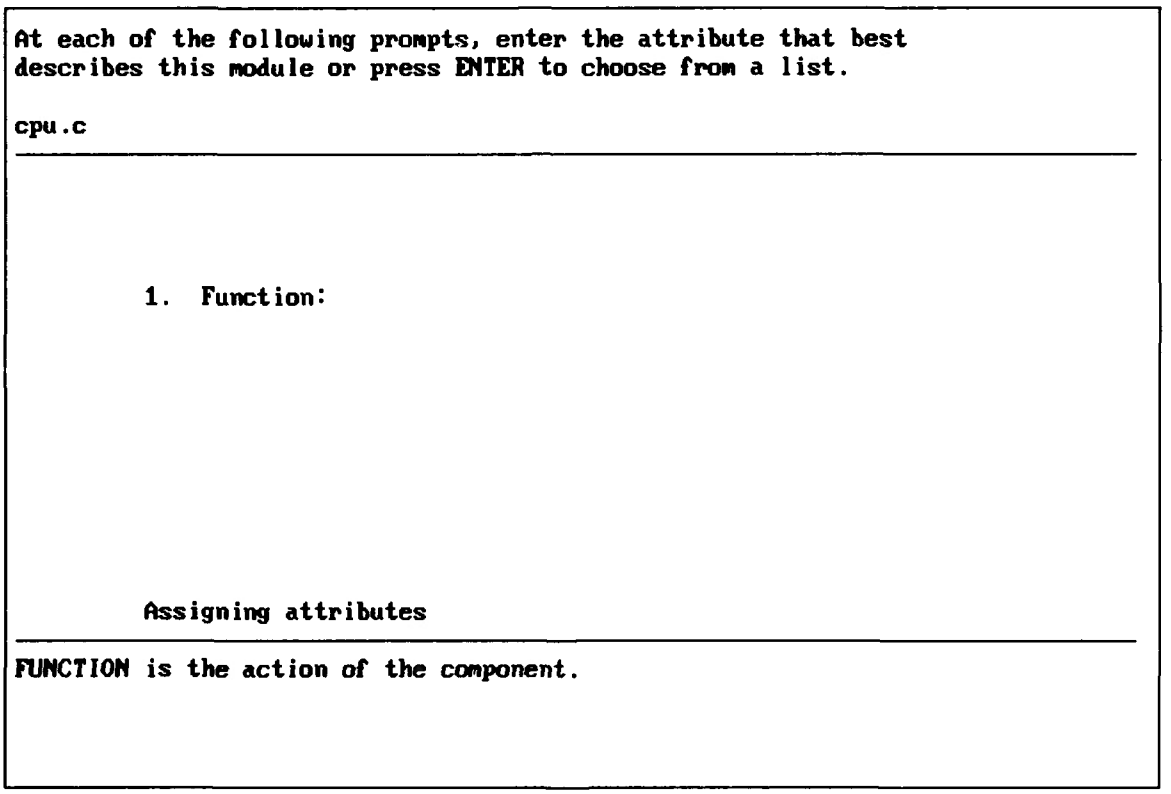
At the opening menu, the user is presented with three choices as follows.

The Opening Menu



Choice A is for adding a component to the repository. The prompts are self-explanatory. Next the user is prompted for the component name; the system must be able to find the component before it can be added to the repository. When the system is ready for the user to assign attributes to the module, the user will see the following screen.

Attribute Assignment Screen



The user enters the attributes as prompted. After all the descriptors have been input, the user can make changes. When the descriptors are accepted, the component will be

added to the system. No maintenance is required for this phase of the system.

3. Making Queries of the System

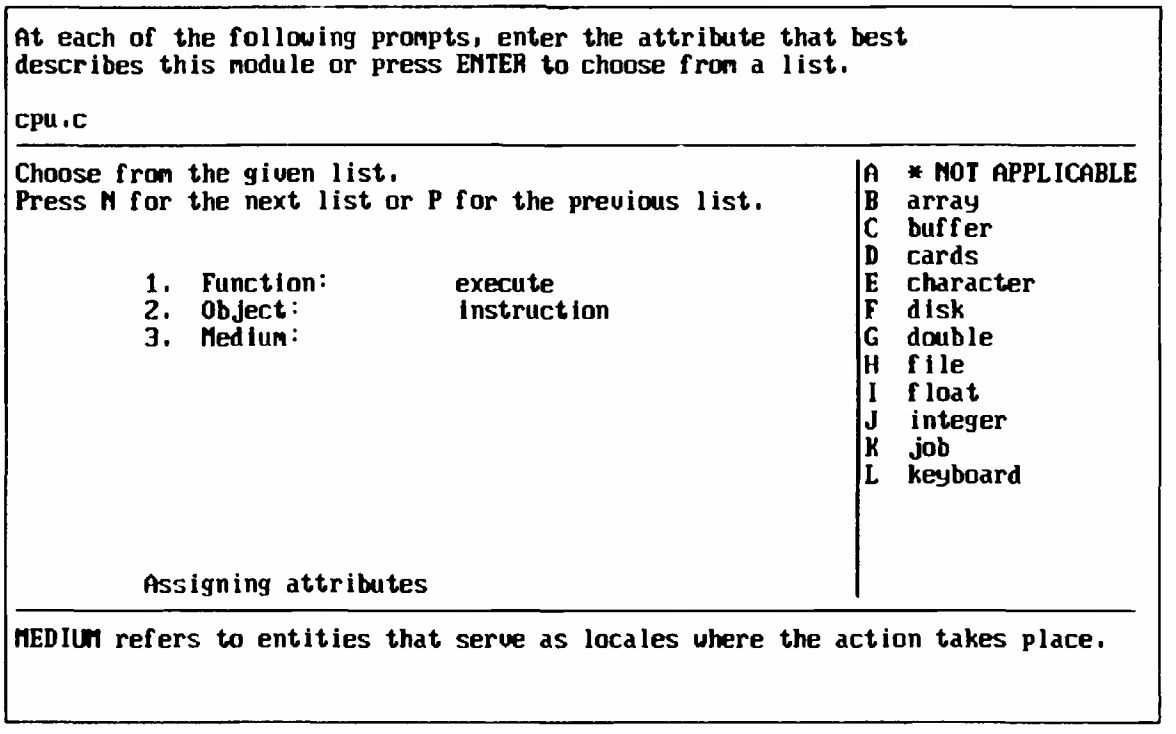
Once the user chooses Q at the opening menu, the system updates the Software Attribute Database (SADB) to insure that the additions are included. The entering of attributes for a query is done exactly as assigning attributes to a new component. When the list of desired attributes has been accepted, SADB is searched for all components that meet one or more of the desired attributes. All exact matches are displayed automatically. Subsequently, the user is presented with a message at the bottom of the screen containing a component name and the percentage of the attributes matched. The user can make the decision to view a component based on the number of attributes matched.

After all the possible candidates have been view or passed by, the user is asked if (s)he wants to keep the list of candidates. If a negative response is given, the list is discarded and the program returns to the main menu. If accepted, the user is prompted for a file name and up to two lines of comments. Subsequently, the candidates with their attributes are dumped into the file. While viewing the result of a system query, the user can press D and the system will display the facet definition at the bottom of the screen.

4. Using the Thesaurus

During the assigning of attributes and while making queries of the system, the thesaurus is available by pressing the ENTER key. When the key is pressed, a list of possible choices for the current facet is displayed on the right side of the screen. Also, if the user enters a descriptor not included in the common vocabulary, a list will appear on the right side of the screen. By selecting a letter corresponding to one of the given choices, the program inserts the selected attribute. N and P are presented to get the next and previous lists.

Facet Options for the Attributes



5. Updating SADB

NOTE: The user cannot make any changes to the software repository from inside the program except for addition of modules. It is recommended that the user not make any deletions from SADB.

For deletion of components from SADB, use any ASCII text editor. Each new line in the data file named ru.dta is a set of component descriptors. The line is a comma-delimited list containing the component name, function, object, medium, system type, functional area, setting, and language in that order. A ninth entry may be present if the user has assigned a custom attribute from a user supplied thesaurus file.

To delete a component from SADB, find the line containing its name and delete it. The next time the system is used to make a query, the SADB supporting files will be updated automatically.

6. Updating the Thesaurus/Common Vocabulary

NOTE: It is recommended that the user not be allowed to make changes to the thesaurus data files to avoid compromising the integrity of the supporting thesaurus.

To update one of the common vocabulary lists, use any ASCII text editor. The thesaurus entries are stored in a comma-delimited list with the first word of the list being the primary key for that particular set of descriptors. Each line of descriptors must be terminated with a carriage

return. It is recommended that the System Administrator perform all additions and deletions. After changes are made, at least one of the supporting files for the particular facet must be deleted. The next time the thesaurus accesses the list for the updated facet, its supporting files will be updated. The facets used for this program and the associated file names of their respective thesauruses are:

function - funct.ths
object - object.ths
medium - medium.ths
system type - systype.ths
functional area - funcarea.ths
setting - setting.ths

A thesaurus file for language (lang.ths) is included for information only; language is not used as a parameter during queries.

NAMES OF THE DATA FILES WITH THEIR SUPPORTING FILES

Data File	Wrd File	Inverted List	Inverted List Index
ru.dta	ru.wrd	ru.inv	ru.vdx
funct.ths	funct.wrd	funct.inv	funct.vdx
object.ths	object.wrd	object.inv	object.vdx
medium.ths	medium.wrd	medium.inv	medium.vdx
systype.ths	systype.wrd	systype.inv	systype.vdx
funcarea.ths	funcarea.wrd	funcarea.inv	funcarea.vdx
setting.ths	setting.wrd	setting.inv	setting.vdx
lang.ths	lang.wrd	lang.inv	lang.vdx

Below are the purposes and contents of the supporting files.

.wrd - contains all words in the data file with their line numbers that are used to create the inverted list

.inv - an inverted list for the data file

.vdx - the index file for the inverted list

APPENDIX C

PROGRAM SOURCE CODE LISTING

```
/*
* PROGRAMMER:  JOE E. SWANSON
* [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

/* ru.h */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <graph.h>

enum color {
        Black,
        Blue,
        Green,
        Cyan,
        Red,
        Magenta,
        Brown,
        White,
        Dgray,
        Light_blue,
        Lgreen,
        Lcyan,
        Lred,
        Light_magenta,
        Yellow,
        Bwhite,
        };

enum {
        Buf_size =  257,          /* buffer size */
        FW       =  25,           /* field width */
        LEN      =  81,
        Required =   1,
        Optional =   2,
        };

typedef enum { FALSE,TRUE } BOOLEAN;

void screens(int screen);

#define cls               _clearscreen (_GCLEARSCREEN)
#define clw               _clearscreen (_GWINDOW)
#define gotoxy            _settextposition
#define _ot               _outtext

#define window0           _settextwindow(1,1,25,80); \
                           _settextcolor(White)
```

```
#define menuwindow          screens(9); _settextwindow(7,15,18,65);\
                            _settextcolor(White)
#define windowmsg           screens(8); _settextwindow(23,1,25,80);\
                            _settextcolor(Cyan)

#define window1             screens(1); _settextwindow(1,1,4,80);  \
                            _settextcolor(White)
#define window2            _settextwindow(6,1,8,59); \
                            _settextcolor(Light_magenta)
#define window3            _settextwindow(10,10,21,59); \
                            _settextcolor(Yellow)
#define windowthes          screens(7); _settextwindow(6,61,21,80); \
                            _settextcolor(Bwhite)

#define window4            _settextwindow(6,1,21,18); \
                            _settextcolor(Lcyan)
#define window5            _settextwindow(6,19,21,38); \
                            _settextcolor(Lred)
#define window6            _settextwindow(6,39,21,59); \
                            _settextcolor(White)

#define Read               "r+"
#define Write              "w+"
#define Append             "a+"
#define Hline               "————————————————————————————————\
————————————————————————————————————————————————"

typedef struct invert {
          char key[FW];
          long offset;
          } INVERTNDX;

typedef struct component {
          char comp_name[LEN],    /* component name */
                 function[LEN,
               object[LEN],
               medium[LEN],
               system_type[LEN],
               funct_area[LEN],
               setting[LEN],
               language[LEN],
               other[LEN];
               } COMPONENT;

BOOLEAN chg_attribute(COMPONENT *,char);
BOOLEAN respond(void);

char *sgets(char *string,int length,char *stream);

int check_path(char input[], char varname[], int flag);

void attr_def(int);
```

```
void get_attr(int row,char attr[],char facet[],char thesdta[]);
void hyper(char [], char []);
void open_file(FILE **, char [], char [], char []);
void query(char ifname[],char invfile[],char vdxfile[]);
void sort(char input[]);
void thesaurus(char code,char key[],char dtafile[]);
void trap(int, char [], char []);
void unique(char file_name[]);

char otext[LEN];                    /* global string for outtext */
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

#include "ru.h"
#include <conio.h>

#define         cbuffer         10001

                                  /* FUNCTION PROTOTYPES */
BOOLEAN chg_attribute(COMPONENT *,char);
BOOLEAN does_exist(char []);
BOOLEAN respond(void);

char menu(void);
char *read_attr(char [], char []);

COMPONENT *read_attributes(char *, COMPONENT *);

void add_component(void);
void assign_attributes(char *, char *);
void attr_def(int);
void db_update(void);
void get_attr(int row,char attr[],char facet[],char thesdta[]);
void get_other_filename(char name[LEN]);
void invert(char inputfile[],char invfile[],char vdxfile[]);
void thesaurus(char code,char key[],char dtafile[]);

/*************************************************************************
*******************************  MAIN  ***********************************
*************************************************************************
*
* This program prompts the user for an input file.  If the file exists,
* then the user is asked if the input file has been assigned attributes.
* If a negative response is given, the user is prompted for the
* attributes, else the module is added to the system.
*
*************************************************************************/
void main(void)
{
  BOOLEAN add_flag = TRUE;        /* set when component added */

  cls;
  _wrapon (_GWRAPOFF);            /* truncate at window's edge */
                                  /* look for DOS sort.exe */
  check_path("sort.exe","PATH",1);
  /* if no components have been previously added go directly to
   * add_component */
```

```
if (!check_path("ru.dta","",Optional))
{
  _ot("The reuse database is empty.");
  _ot("  Do you wish to add components? [Y/N] ");
  gotoxy(1,70);
  if(respond()) add_component();
  else exit(0);
}

if (check_path("ru.dta","",Optional));
else
{                                  /* components have not been added */
  cls;
  exit(0);
}

while (TRUE)
{
  cls;

  switch(menu())
  {
    case 'A':
      cls;
      add_flag = TRUE;
      add_component();
      cls;
      break;

    case 'E':
      cls;
      window2;
      _ot("Do you want to exit to the operating system?  [Y/N] ");
      gotoxy(1,53);
      if (respond())
      {
        window0;
        cls;
        exit(0);
      }
      break;

    case 'Q':
      cls;
      if (add_flag)          /* CREATE/UPDATE supporting files */
      {
        db_update();
        add_flag = FALSE;
      }
                               /* begin processing the query */
      query("ru.dta", "ru.inv", "ru.vdx");
      break;
```

```
      default: break;
    } /* switch */
  } /* while */
}  /* main

*****************************************************************************/

/*****************************************************************************
************************  chg_attribute  *************************************
*****************************************************************************
*
* This function is called if an entered attribute is to be changed.
*
*****************************************************************************/
BOOLEAN chg_attribute(COMPONENT *current,char flag)
{
  char buffer[LEN];
  int x;

  window1;
  clw;
  windowmsg;
  clw;
  window3;
  gotoxy(9,1);
  _ot("Are the above attributes correct?  [Y/N] ");
  gotoxy(9,42);
  if (respond())
    x = FALSE;
  else
  {
    window1;
    _ot("Enter the attribute that best describes this facet of the ");
    _ot("component.");
    window3;
    gotoxy(11,1);
    _ot("Enter the number of the attribute to change -- ");
    buffer[0] = (char) getche();
    gotoxy(11,1);
    _ot("                                                  ");
    x = atoi(buffer);
    gotoxy(x,1);
    _ot("                                                   ");
    switch(x)
    {
      case 1:
        get_attr(1,current->function,"Function:","funct.ths");
        break;

      case 2:
        get_attr(2,current->object,"Object:","object.ths");
```

```
        break;

      case 3:
        get_attr(3,current->medium,"Medium:","medium.ths");
        break;

      case 4:
        get_attr(4,current->system_type,"System Type:","systype.ths");
        break;

      case 5:
        get_attr(5,current->funct_area,"Functional Area:",
                                                    "funcarea.ths");
        break;

      case 6:
        get_attr(6,current->setting,"Setting:","setting.ths");
        break;

      case 7:
        if (flag == 'a')
          get_attr(7,current->language,"Language:","lang.ths");
        break;

      case 8:
        buffer[0] = '\0';
        get_attr(8,current->other,"Other:",buffer);

      default: break;
    } /* switch */

    x = TRUE;

  } /* else */

  return x;

}  /* change attribute

*****************************************************************************/
```

```
/*************************************************************************
*************************  does exist  ***********************************
*************************************************************************
*
* This function does a linear search through the software attribute
* database to determine if the inputfile previously existed or if an
* identical file name is found.
*
*************************************************************************/
BOOLEAN does_exist(char fname[])
{
  BOOLEAN flag = FALSE;
  char *ptr, line[Buf_size];
  FILE *fp;

  open_file(&fp,Read,"ru.dta","does_exist");

  while (fgets(line,Buf_size,fp) != NULL)
  {
    ptr = strtok(line,",");
    if (strcmp(fname,ptr) == 0)
    {
      window1;
      clw;
      sprintf(otext,"Component %s previously added.  SKIPPING!\n",
                                                          fname);
      _ot(otext);
      sprintf(otext,"press any key to continue...");
      _ot(otext);
      getch();
      flag = TRUE;
      break;
    }
  }

  fclose(fp);
  return flag;

}  /* does exist

*************************************************************************/
```

```
/*************************************************************************
*************************  respond  **************************************
*************************************************************************
*
* This function prompts the user for a Y or N response.
*
*************************************************************************/
BOOLEAN respond(void)
{
  char x;
  struct rccoord oldpos;  /* catch current cursor position */

  oldpos = _gettextposition();

  do
  {
    x = (char) toupper(getche());
    gotoxy(oldpos.row,oldpos.col);
  }
  while ((x) != 'N' && (x) != 'Y');

  return (x == 'Y') ? TRUE : FALSE;

}  /* respond

*************************************************************************/

/*************************************************************************
*************************  menu  ****************************************
*************************************************************************
*
* This function places a menu on the screen.  The users choice is
* returned.
*
*************************************************************************/
char menu(void)
{
  menuwindow;

  _ot("Software Repository.");
  gotoxy(5,1);
  _ot("(A)dd a component to the system.");

  gotoxy(7,1);
  _ot("(Q)uery the system for a component.");

  gotoxy(9,1);
  _ot("(E)xit to DOS.");

  gotoxy(12,1);
  _ot("Make your selection -- ");
  return (char) toupper(getche());
```

```
}  /* menu

*****************************************************************************/

/*****************************************************************************
***************************  read_attr  ***************************************
*****************************************************************************
*
* This function reads one attribute from the buffer.  The attribute is
* returned to the calling module.
*
*****************************************************************************/
char *read_attr(char buffer[],char sep[])
{
  char input[FW],
       *pchar,
       ch;

  if (buffer[0] != ',')
  {
    pchar = strstr(buffer,sep);
    pchar++;
  }
  else
  {
    pchar = strstr(buffer,",");
    pchar++;
  }
                                    /* read data */
  if (buffer[0] != ',')
    strcpy(input,strtok(buffer,sep));
  else
    strcpy(input," ");

  strcpy(buffer,pchar);

  return input;

}  /* read attr

*****************************************************************************/
```

```
/*************************************************************************
************************  read attributes  ******************************
*************************************************************************
*
* This function reads the attributes from the beginning of the input
* file.  The attribute data file is checked, if the file is not
* currently in the database, it will be added.  The attributes are
* passed back via the formal parameter list.
*
*************************************************************************/
COMPONENT *read_attributes(char *fname,COMPONENT *new)
{
  static char buffer[Buf_size],
              *pchar,             /* char pointer */
              *temp;
  char ch = '\0';
  int x,                  /* number of bytes read by fread */
      i;
  FILE *fp;

  open_file(&fp,Read,fname,"read_attributes");
  x = fread (buffer,1,Buf_size-1,fp);
  fclose(fp);

  pchar = strstr(buffer," ");      /* remove opening comment */
  pchar++;
  strcpy(buffer,pchar);
                                   /* read component name */
  for (i = 0; i < 30; i++) new->comp_name[i] = buffer[i];

  temp = strtok(new->comp_name,",");
                                   /* check to see if exist in module */
  pchar = strstr(fname,temp);

  if (pchar != NULL)
  {
    strcpy(new->comp_name,read_attr(buffer,","));
    strcpy(new->function,read_attr(buffer,","));
    strcpy(new->object,read_attr(buffer,","));
    strcpy(new->medium,read_attr(buffer,","));
    strcpy(new->system_type,read_attr(buffer,","));
    strcpy(new->funct_area,read_attr(buffer,","));
    strcpy(new->setting,read_attr(buffer,","));
    strcpy(new->language,read_attr(buffer,","));
    strcpy(new->other,read_attr(buffer,","));
  }
  else
    {
      window1;
      clw;
      _ot("The specified module does not contain parameters.\n");
      new = NULL;
```

```
    }

  return new;

}  /* read attributes

*****************************************************************************/

/*****************************************************************************
************************  add_component  ************************************
*****************************************************************************
*
* This function is called when a component is being added to the system.
* It calls the necessary functions for attribute assignment, reading of
* the attributes from the module, and adding the new info to the system
* information.
*
*****************************************************************************/
void add_component(void)
{
  BOOLEAN exist;
  char ifname[LEN],
       *pchar,
       path[LEN], drive[3], dir[3], fname[9], ext[5],
       added[Buf_size] = "";
  static char buffer[cbuffer];
  COMPONENT current,
            *curr = NULL;
  FILE *fp;
  int x, i = 0;
  static int col = 1, row = 1;

  while (TRUE)
  {
    window1;
    clw;
                                    /* prompt user for file name */
    gotoxy(1,1);
    _ot("Enter the name of the new component:  ");
    fflush(stdin);
    gets(path);
    strlwr(path);          /* change to lowercase */

                           /* change \ to / to avoid escape char in path */
    while ((pchar = strstr(path,"\\")) != NULL)
      pchar[0] = '/';

    while((fp = fopen(path,Read)) == NULL)
    {
      window1;
      clw;
      _ot("New component cannot be found.  Enter new component \
```

```
name.\n");
      _ot("(You may need to include the correct path. (Q)uit)");
      gotoxy(1,60);
      fflush(stdin);
      gets(path);
      strlwr(path);
                          /* change \ to / to avoid escape char in path */
      while ((pchar = strstr(path,"\\")) != NULL)
        pchar[0] = '/';

      if (path[0] == 'q')
        return;
    }

    fcloseall();
    _splitpath(path, drive, dir, fname, ext);
    strcpy(ifname,fname);
    strcat(ifname,ext);

    if ((exist = does_exist(ifname)) == FALSE)
    {
      assign_attributes(path, ifname);
      curr = read_attributes(ifname,&current);
    }

    if (!exist && curr != NULL)
    {
      open_file(&fp,"a+","ru.dta","read_attributes");
      fprintf(fp,"%s,%s,%s,%s,%s,%s,%s,%s,%s\n",current.comp_name,
              current.function,current.object,current.medium,
              current.system_type,current.funct_area,current.setting,
              current.language,current.other);
      fclose(fp);
      window2;
      gotoxy(1,1);
      sprintf(otext,"%-14.14s ",ifname);
      strcat(added,otext);
      _ot(added);
      col += 15;
      if(col >= 45)
      {
        strcat(added,"\n");
        col = 0;
      }

      if (strlen(added) > 165)
        added[0] = '\0';
    }

    window1;
    clw;
    _ot("Do you wish to add another component?  [Y/N] ");
```

```
    gotoxy(1,47);
    if (!respond()) break;
  }

}  /* add component

*****************************************************************************/

/*****************************************************************************
************************  assign attributes  ***************************
*****************************************************************************
*
* This function takes a module that has not been assigned attributes and
* prompts the user for the appropriate attribute definitions.  The
* attributes are appended to the beginning of the input file.
*
*****************************************************************************/
void assign_attributes(char *path, char *inputfile)
{
  COMPONENT current;
  static char buffer[cbuffer],
              *pchar,
             newfile[LEN] = "";
  FILE *fp, *ifp;
  int x;

  window1;
  clw;
  _ot("At each of the following prompts, enter the attribute that \
best\n");
  _ot("describes this module or press ENTER to choose from a list.");
  gotoxy(4,1);
  _ot(inputfile);

  window3;
  gotoxy(21,1);
  _ot("Assigning attributes");

  get_attr(1,current.function,"Function:","funct.ths");
  get_attr(2,current.object,"Object:","object.ths");
  get_attr(3,current.medium,"Medium:","medium.ths");
  get_attr(4,current.system_type,"System Type:","systype.ths");
  get_attr(5,current.funct_area,"Functional Area:","funcarea.ths");
  get_attr(6,current.setting,"Setting:","setting.ths");
  get_attr(7,current.language,"Language:","lang.ths");
  get_attr(8,current.other,"Other:",newfile);

  window1;
  clw;
  while (chg_attribute(&current,'a'));
  clw;
```

```
  open_file(&fp,Write,"work","assign_attributes");
  fprintf(fp,"/* %s,%s,%s,%s,%s,%s,%s,%s,%s, */\n",inputfile,
          current.function,current.object,current.medium,
          current.system_type,current.funct_area,current.setting,
          current.language,current.other);

  open_file(&ifp,Read,path,"assign_attributes");

  while((x = fread(buffer,1,cbuffer,ifp)) != 0)
    fwrite(buffer,x,1,fp);

  fclose(ifp);
  fclose(fp);

  if (strcmp(path,inputfile) == 0)
    unlink(inputfile);

  rename ("work", inputfile);

}  /* assign attributes

*****************************************************************************/

/*****************************************************************************
**************************  open_file  ***************************************
*****************************************************************************
*
*  This procedure opens the input and output files for this program.
*
*****************************************************************************/
void open_file(FILE **fp,char type[],char name[],char module[])
{
  /*  open input and output files  */
  if ((*fp = fopen(name,type)) == NULL)
  {
    window0;
    cls;
    trap(1001,name,module);
    exit(1001);
  }
}  /* open_file */

/*****************************************************************************/
```

```
/*************************************************************************
**************************  ERROR TRAPS  *********************************
*************************************************************************
*
* This function produces an error message when the input string does
* not match one of those specified in the program specification.  This
* function is called by modules listed in the error messages.
*
*************************************************************************/
void trap(int code, char filename[], char module[])
{
  fprintf(stderr,"\n==> ERROR\n");

  switch(code)
  {
    case 1001 :  fprintf(stderr,"INPUT/OUTPUT FILE NOT \
FOUND(IO 1001).\n");
                 fprintf(stderr,"File \"%s\" not found in function \
\"%s.\"\n",filename,module);
                 break;

    case 2001 :  fprintf(stderr,"\nFile not properly sorted(%s2001).\
\n\n",module);
                 break;
/* not used yet
    case 3001 :  fprintf(stderr,"\n (3001).\n\n");
                 break;

    case 3002 :  fprintf(stderr,"\n (3002).\n\n");
                 break;

    case 4001 :  fprintf(stderr,"\n (4001).\n\n");
                 break;

    case 4002 :  fprintf(stderr,"\n (4002).\n\n");
                 break;

    case 5001 :  fprintf(stderr,"\n (5001).\n\n");
                 break;
*/
    default :  fprintf(stderr,"*********  FATAL ERROR  ********\n\n\n");
               exit(999);
               break;

  } /*  switch  */

}  /* trap  */

/*************************************************************************/
```

```
/*************************************************************************
**********************  attribute definitions  ***************************
*************************************************************************
*
* This function lists the definitions of the respective attributes.
*
*************************************************************************/
void attr_def(int n)
{
  windowmsg;
  clw;

  switch(n)
  {
    case 1:
      _ot("FUNCTION is the action of the component.\n");
      break;

    case 2:
      _ot("OBJECT is the object manipulated by the component.\n");
      break;

    case 3:
      _ot("MEDIUM refers to entities that serve as locales where ");
      _ot("the action takes place.\n");
      break;

    case 4:
      _ot("SYSTEM TYPE refers to functionally identifiable, ");
      _ot("application-independent\n");
      _ot("modules.  Usually includes > 1 component.\n");
      break;

    case 5:
         _ot("FUNCTIONAL AREA describes application-dependent \
activities.\n");
      break;

    case 6:
      _ot("SETTING describes where the component is exercised.\n");
      break;

    case 8:
      _ot("You may enter a user defined facet now.");
      break;

    default: break;
  } /* switch */
  window3;
}  /* attr_def

*************************************************************************/
```

```
/*************************************************************************
**************************  get attr  ************************************
*************************************************************************
*
* This function prompts the user and returns the value input.  All
* modifications to the input value are completed before it is returned.
*
*************************************************************************/
void get_attr(int row,char attr[],char facet[],char thesdta[])
{
  attr_def(row);
  window3;
  gotoxy(row,1);
  sprintf(otext,"%i.  %-18s",row,facet);
  _ot(otext);
  fflush(stdin);
  gets(attr);
  attr[FW] = '\0';
  strlwr(attr);
  if (row == 8 && attr[0] != '\0' && attr[0] != '*')
    get_other_filename(thesdta);      /* get other definitions */

  if (thesdta[0] != '\0') /* if file !exist, return */
    thesaurus('s',attr,thesdta);
  else attr[0] = '\0';

  attr_def(row);
  gotoxy(row,1);
  _ot("                                                          ");
  gotoxy(row,1);
  sprintf(otext,"%i.  %-18s%-25s",row,facet,attr);
  _ot(otext);

}  /* get attr

*************************************************************************/

/*************************************************************************
*****************  software attribute database update  *****************
*************************************************************************
*
* This function updates/creates the supporting database file for the
* software attributes.
*
*************************************************************************/
void db_update(void)
{
  unlink("ru.wrd");
  unlink("ru.inv");
  unlink("ru.vdx");
  sort("ru.dta");
  hyper("ru.dta","ru.wrd");
```

```
  invert("ru.wrd","ru.inv","ru.vdx");

}  /* database update

*****************************************************************************/

/*****************************************************************************
**********************  get other filename  ***********************************
*****************************************************************************
*
* This function is used to get the definition file for a user-defined
* facet.  If the file does not exist, NULL is returned.
*
*****************************************************************************/
void get_other_filename(char name[LEN])
{
  windowmsg;
  clw;
  _ot("Enter the name containing the facet definitions -- \n");
  gets(name);
  if (!check_path(name,"",Optional))
    name[0] = '\0';
  clw;
}  /* get other filename

*****************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

#include "ru.h"
#include <conio.h>
                                    /* FUNCTION PROTOTYPES */
BOOLEAN evaluate_targets(char tgtfile[],COMPONENT current);
BOOLEAN find_targets(char ifname[],char invfile[],char vdxfile[],
                                      char tgtfile[],COMPONENT current);
BOOLEAN get_tgts(char ifname[],char invfile[],char vdxfile[],
                  char tgtfile[],char facet[],BOOLEAN flag);

INVERTNDX bin_search(char inputfile[],char key[],int count);
INVERTNDX find_key(char vdxfile[],char key[]);

void display_component(COMPONENT target);
void display_targets(char tgtfile[],COMPONENT user);
void extract(COMPONENT *target, int *hits, char input[]);
void output_facet(int row, char *str);
void output_target(char [], char [], char [], INVERTNDX);
void make_query(COMPONENT *);
void tgttext(char tgtfile[], COMPONENT user);
void query(char ifname[],char invfile[],char vdxfile[]);

/*M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M */
void query(char ifname[],char invfile[],char vdxfile[])
{
  char name[FW],
       tgtfile[FW] = "ru.tgt";
  COMPONENT current;

  cls;
  make_query(&current);
  window0;
  clw;
  gotoxy(10,30);
  _ot("Searching...");

  if (find_targets(ifname,invfile,vdxfile,tgtfile,current))
  {
    evaluate_targets(tgtfile,current);
    display_targets(tgtfile,current);
    tgttext(tgtfile,current);
  } /* if find */

} /* query */

/*m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m */
```

```
/*************************************************************************
************************  evaluate targets  ***************************
*************************************************************************
*
* This function reads each of the targets from the tgt file and insures
* that a match exists in the parameter list and the user list.  Targets
* that do not match are deleted from the list.  If all targets are
* deleted, a False flag is returned and the processing is halted for
* this search.  The language attribute is for info only.  A component in
* the same language does not constitute a match.
*
*************************************************************************/
BOOLEAN evaluate_targets(char tgtfile[],COMPONENT current)
{
  BOOLEAN flag = FALSE;
  char input[Buf_size],
       *pchar,
       seps[FW] = ",\n\r",
       str[FW],
       *duplicate,
       command[LEN] = "sort /r < new$.tgt > out";
  FILE *fp,*fpo;
  int att_matches = 0;

  open_file(&fp,Read,tgtfile,"evaluate_targets");
  while(((pchar = fgets(input,Buf_size,fp)) != NULL) && pchar[0] != '\0'
          && pchar[0] != '\n')
  {
    duplicate = strdup(input);  /* create a duplicate of the input */

    strtok(input,seps);          /* throw away the component name */

    pchar = strtok(NULL,seps);    /* read & compare the function */
    if (strcmp(current.function,pchar) == 0)
      att_matches++;
                                 /* read & compare the object */
    pchar = strtok(NULL,seps);
    if (strcmp(current.object,pchar) == 0)
      att_matches++;
                                 /* read & compare the medium */
    pchar = strtok(NULL,seps);
    if (strcmp(current.medium,pchar) == 0)
      att_matches++;
                                  /* read & compare the sys type */
    pchar = strtok(NULL,seps);
    if (strcmp(current.system_type,pchar) == 0)
      att_matches++;
                                 /* read & compare the funct_area */
    pchar = strtok(NULL,seps);
    if (strcmp(current.funct_area,pchar) == 0)
      att_matches++;
                                      /* read & compare the setting */
```

```
    pchar = strtok(NULL,seps);
    if (strcmp(current.setting,pchar) == 0)
      att_matches++;

    pchar = strtok(NULL,seps);  /* discard language */
    pchar = strtok(NULL,seps);  /* read & compare other */
    if (pchar != NULL && strcmp(current.other,pchar) == 0)
      att_matches++;

    if (att_matches)
    {
      flag = TRUE;
      open_file(&fpo,Append,"new$.tgt","evaluate targets");
      itoa(att_matches,str,10);
      fprintf(fpo,"%s,%s",str,duplicate);
      fclose(fpo);
    }

    free(duplicate);
    att_matches = 0;
  } /* while */

  fclose(fp);
  system(command);
  unlink("new$.tgt");
  unlink(tgtfile);
  rename("out", tgtfile);
  return flag;

}  /* evaluate targets

*****************************************************************************/

/*****************************************************************************
************************  find targets  ***********************************
*****************************************************************************
*
* This file takes the current set of component attributes and locates
* all components that match n or more attributes.  The target list is
* put in file "ru.tgt."
*
*****************************************************************************/
BOOLEAN find_targets(char ifname[],char invfile[],char vdxfile[],
                                        char tgtfile[],COMPONENT current)
{
  BOOLEAN flag = FALSE;           /* True indicates target found */

  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.function,
                                                          flag);
  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.object,flag);

  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.medium,flag);
```

```
  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.system_type,
                                                              flag);
  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.funct_area,
                                                              flag);
  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.setting,
                                                          flag);
  flag = get_tgts(ifname,invfile,vdxfile,tgtfile,current.other,flag);

  if (flag)
  {
    sort(tgtfile);
    unique(tgtfile);
  }
  else
  {
    clw;
    _ot("No targets found.\n\nPress any key to continue...");
    getch();
  }
  return flag;
}  /* find targets

*************************************************************************/

/*************************************************************************
************************  binary search  ********************************
*************************************************************************
*
* This function performs a binary search of data stored on secondary
* storage.  It returns the index or NULL if not found.
*
*************************************************************************/
INVERTNDX bin_search(char inputfile[],char key[],int count)
{
  FILE *fp;
  int low = 0, high = count-1,
      test;
  INVERTNDX index;
  long guess, look;

  open_file(&fp,"rb",inputfile,"bin_search");
  while(low <= high)
  {
    guess = (long) (low + high)/2;        /* find mid_point */
    look = guess * sizeof(INVERTNDX);
    fseek(fp,look,SEEK_SET);
    fread(&index,1,sizeof(INVERTNDX),fp);

    if ((test = strcmp(key,index.key)) < 0)
      high = (int) guess - 1;
    else
      if (test > 0)
```

```
        low = (int) guess + 1;
      else
      {
        fclose(fp);
        return index;
      }

  } /* while low */

  fclose(fp);
  strcpy(index.key,"");
  return index;                /* key not found */

}  /* bin search

*****************************************************************************/

/*****************************************************************************
*****************************  find_key  *************************************
*****************************************************************************
*
* This function finds the specified key, if it exists, on secondary
* storage.
*
*****************************************************************************/
INVERTNDX find_key(char vdxfile[],char key[])
{
  FILE *fp;
  int count;                        /* number of elements in vdxfile */
  INVERTNDX index;
  long pos;                        /* number of bytes in file */
                                      /* find number of index entries */
  open_file(&fp,"rb",vdxfile,"find key");
  fseek(fp,0L,SEEK_END);
  pos = ftell(fp);                  /* get file length */
  fclose(fp);
  count = (int) pos / sizeof(INVERTNDX);
  index = bin_search(vdxfile,key,count);
  return index;
}  /* find key */

/*****************************************************************************/
```

```
/*****************************************************************************
************************  display targets  ***********************************
*****************************************************************************
*
* This function displays on the screen all the targets found during the
* search.
*
*****************************************************************************/
void display_targets(char tgtfile[],COMPONENT user)
{
  char input[Buf_size],
       ch;
  COMPONENT target;
  FILE *fp;
  int num,
      hits,                        /* number of parameter matches */
      wronganswer = TRUE;
  float request = 0;
                                   /* create screen display */
  cls;
  window4;
  gotoxy(1,4);
  _ot("FACETS");

  gotoxy(3,4);
  _ot("Component:");

  gotoxy(4,1);
  _ot("1. Function:");

  gotoxy(6,1);
  _ot("2. Object:");

  gotoxy(7,1);
  _ot("3. Medium:");

  gotoxy(9,1);
  _ot("4. System type:");

  gotoxy(10,1);
  _ot("5. Funct. area:");

  gotoxy(12,1);
  _ot("6. Setting:");

  gotoxy(13,1);
  _ot("7. Language:");

  gotoxy(15,1);
  _ot("8. Other:");

  window5;                             /* display user request */
```

```
gotoxy(1,1);
_ot("USER'S REQUEST");

gotoxy(4,1);
sprintf(otext,"%-.*s",18,user.function);
_ot(otext);

gotoxy(6,1);
sprintf(otext,"%-.*s",18,user.object);
_ot(otext);

gotoxy(7,1);
sprintf(otext,"%-.*s",18,user.medium);
_ot(otext);

gotoxy(9,1);
sprintf(otext,"%-.*s",18,user.system_type);
_ot(otext);

gotoxy(10,1);
sprintf(otext,"%-.*s",18,user.funct_area);
_ot(otext);

gotoxy(12,1);
sprintf(otext,"%-.*s",18,user.setting);
_ot(otext);

gotoxy(15,1);
sprintf(otext,"%-.*s",18,user.other);
_ot(otext);
                                /* get number of user attributes input */
if (user.function[0] != '\0') request++;
if (user.object[0] != '\0') request++;
if (user.medium[0] != '\0') request++;
if (user.system_type[0] != '\0') request++;
if (user.funct_area[0] != '\0') request++;
if (user.setting[0] != '\0') request++;
if (user.other[0] != '\0') request++;
                                /* display targets */
open_file(&fp,Read,tgtfile,"display targets");
while(fgets(input,Buf_size,fp) != NULL)
{
  extract(&target,&hits,input);
  if (hits / request == 1) display_component(target);
  else
  {
    windowmsg;
    clw;
    sprintf(otext,"Component %s meets %5.2f %% of the request.\n",
            target.comp_name,hits / request * 100);
    _ot(otext);
    _ot("Display -- [Y/N] ");
```

```
  if (respond())
  {
    clw;
    display_component(target);
  }
  else continue;
}

while (wronganswer)
{
  window1;
  clw;
  if (hits > 1)
  {
    sprintf(otext,"%2i attributes matched.\n",hits);
    _ot(otext);
  }
  else
  {
    sprintf(otext,"%2i attribute matched.\n",hits);
    _ot(otext);
  }

  _ot("Press (C)ontinue, Facet (D)efinitions, or (Q)uit... ");
  ch = (char) tolower(getch());

  switch(ch)
  {
    case 'c':
      wronganswer = FALSE;
      break;

    case 'd':
      windowthes;
      _ot("Enter the number of\nthe facet or\n0 to return\n");
      while((num = getche() - 48) != 0)
      {
        if (num != 8)
          attr_def(num);
        windowthes;
        gotoxy(4,1);
      }  /* while */
      clw;
      break;

    case 'q':
      wronganswer = FALSE;
      break;

    default: break;

  } /* switch */
```

```
    }  /* while wronganswer */

    wronganswer = TRUE;
    window6;

    if (ch == 'q')
      break;

  } /* while fgets */
  fclose(fp);

} /* display targets
*****************************************************************************/

/*****************************************************************************
*****************************  extract  ******************************
*****************************************************************************
*
* This function is used to break down the tokens of the input string and
* put them in their respective structure members.
*
*****************************************************************************/
void extract(COMPONENT *current, int *hits, char input[])
{
  char str[FW],
       seps[FW] = ",\n\r";

  strcpy(str,strtok(input,seps));  /* read the hits */
  *hits = atoi(str);

  strcpy(current->comp_name,strtok(NULL,seps));
                                   /* read the function */
  strcpy(current->function,strtok(NULL,seps));
                                  /* read the object */
  strcpy(current->object,strtok(NULL,seps));
                                  /* read the medium */
  strcpy(current->medium,strtok(NULL,seps));
                                   /* read the sys type */
  strcpy(current->system_type,strtok(NULL,seps));
                                  /* read the funct_area */
  strcpy(current->funct_area,strtok(NULL,seps));
                                     /* read the setting */
  strcpy(current->setting,strtok(NULL,seps));

  strcpy(current->language,strtok(NULL,seps));

  strcpy(current->other,strtok(NULL,seps));

}  /* extract

*****************************************************************************/
```

```
/*****************************************************************************
*************************  make query  ***************************************
*****************************************************************************
*
* This function allows the user to complete the query questionnaire.
*
*****************************************************************************/
void make_query(COMPONENT *current)
{
  char buffer[LEN] = "";

  window1;
  clw;

  _ot("At each of the following prompts, enter the attribute that \
best\n");
  _ot("describes the desired module or press ENTER to choose from a \
list.");

  window3;
  gotoxy(21,1);
  _ot("Making Query");

  fflush (stdin);

  get_attr(1,current->function,"Function:","funct.ths");
  get_attr(2,current->object,"Object:","object.ths");
  get_attr(3,current->medium,"Medium:","medium.ths");
  get_attr(4,current->system_type,"System Type:","systype.ths");
  get_attr(5,current->funct_area,"Functional Area:","funcarea.ths");
  get_attr(6,current->setting,"Setting:","setting.ths");
  get_attr(8,current->other,"Other:",buffer);

  while (chg_attribute(current,'q'));

  clw;

}  /* make query

*****************************************************************************/
```

```
/*************************************************************************
************************  output target  ********************************
*************************************************************************
*
* This function insures that a target is found.  It then retrieves the
* target info and outputs it to the tgt file.
*
*************************************************************************/
void output_target(char ifname[],char tgtfile[],char invfile[],
                                                    INVERTNDX index)
{
  char list[Buf_size],
       target[Buf_size],
       file[FW],
       *pchar,
       scr[FW];
  FILE *fp;
  int num, i;
                                  /* get the inverted list entry */
  open_file(&fp,"rb",invfile,"output_target");
  fseek(fp,index.offset,SEEK_SET);
  fgets(list,Buf_size,fp);
  fclose(fp);
  strtok(list," ");                 /* remove the words from the list */

  while (TRUE)
  {
    pchar = strtok(NULL," :");
    if (pchar == NULL)
      break;
    /* strcpy(file,pchar);   save for future refinement */
    strcpy(scr,strtok(NULL," :"));
                                  /* retrieve attributes from data file */
    num = atoi(scr);
    open_file(&fp,"r",ifname,"output_target");

    for (i = 1; i <= num; i++)
      fgets(target,Buf_size,fp);

    fp = freopen(tgtfile,Append,fp);
    fputs(target,fp);
    fclose(fp);

  } /* while */

}  /* output target */

/*************************************************************************/
```

```
/*****************************************************************************
************************  target text  ***************************************
*****************************************************************************
*
* This function asks the user whether or not the tgt information is to
* be retained.  If so, the user is asked for a file name and the info is
* put in the specified file, otherwise it is deleted.
*
*****************************************************************************/
void tgttext(char tgtfile[], COMPONENT user)
{
  char name[LEN],
       data[Buf_size];
  COMPONENT curr;
  FILE *fp, *fpo;
  int hits;

  cls;
  window1;

  gotoxy(2,1);
  _ot("Do you wish to keep the target file?  [Y/N] ");
  gotoxy(2,45);
  if (!respond()) unlink(tgtfile);
  else
  {
    clw;
    puts("\nEnter new target file name.");
    gets(name);
    clw;

    while(check_path(name,"",Optional))
    {
      sprintf(otext,"The file, %s, already exist.  Overwrite?  [Y/N]  \
",name);
      _ot(otext);
      if (!respond())
      {
        clw;
        puts("\nEnter new target file name.");
        gets(name);
        clw;
      }
      else
      {
        clw;
        break;
      }
    }

    open_file(&fp,Read,tgtfile,"tgtoutput");
    open_file(&fpo,Write,name,"tgtoutput");
```

```
    _ot("Enter up to two lines of comments.\n");

    gets(data);
    if (data[0] != '\0') fprintf(fpo,"\n%-80s\n",data);

    gets(data);
    if (data[0] != '\0') fprintf(fpo,"%-80s\n",data);

    fprintf(fpo,"\nThe attributes requested are:\n\n");
    fprintf(fpo,"\t%-20s -- %-40s\n","Function",user.function);
    fprintf(fpo,"\t%-20s -- %-40s\n","Object",user.object);
    fprintf(fpo,"\t%-20s -- %-40s\n","Medium",user.medium);
    fprintf(fpo,"\t%-20s -- %-40s\n","System Type",user.system_type);
    fprintf(fpo,"\t%-20s -- %-40s\n","Functional Area",user.funct_area);
    fprintf(fpo,"\t%-20s -- %-40s\n","Setting",user.setting);
    fprintf(fpo,"\t%-20s -- %-40s\n\n","Other",user.other);

    fprintf(fpo,"%s\n\n","****************************************");

    while(fgets(data,Buf_size,fp) != NULL)
    {
      extract(&curr,&hits,data);
      fprintf(fpo,"\nThe attributes of %s are:\n\n",curr.comp_name);
      fprintf(fpo,"\t%-20s -- %-40s\n","Function",curr.function);
      fprintf(fpo,"\t%-20s -- %-40s\n","Object",curr.object);
      fprintf(fpo,"\t%-20s -- %-40s\n","Medium",curr.medium);
      fprintf(fpo,"\t%-20s -- %-40s\n","System Type",curr.system_type);
      fprintf(fpo,"\t%-20s -- %-40s\n","Functional Area",
                                          curr.funct_area);
      fprintf(fpo,"\t%-20s -- %-40s\n","Setting",curr.setting);
      fprintf(fpo,"\t%-20s -- %-40s\n","Language",curr.language);
      fprintf(fpo,"\t%-20s -- %-40s\n\n","Other",curr.other);
      if (hits == 1)
        fprintf(fpo,"%s matched %i attribute.\n",curr.comp_name,hits);
      else
        fprintf(fpo,"%s matched %i attributes.\n",curr.comp_name,hits);

      fprintf(fpo,"%s\n\n","****************************************");
    } /* while */
  } /* else */

  fcloseall();
  unlink(tgtfile);

}  /* target text

*****************************************************************************/
```

```
/*************************************************************************
************************  get targets  ***********************************
*************************************************************************
*
* This function locates all occurrences of the given facet within the
* software attribute database.
*
*************************************************************************/
BOOLEAN get_tgts(char ifname[],char invfile[],char vdxfile[],
                                 char tgtfile[],char facet[],BOOLEAN flag)
{
  INVERTNDX target;
                                 /* find the targets */
  target = find_key(vdxfile,facet);
  if (target.key[0] != '\0')
  {
    output_target(ifname,tgtfile,invfile,target);
    flag = TRUE;
  }

  return flag;

}  /* get targets

*************************************************************************/

/*************************************************************************
************************  display component  ****************************
*************************************************************************
*
* This function displays the given component on the screen alongside the
* user's request.
*
*************************************************************************/
void display_component(COMPONENT target)
{
  window6;
  clw;
  gotoxy(1,1);
  _ot("CANDIDATE");

  output_facet(3,target.comp_name);
  output_facet(4,target.function);
  output_facet(6,target.object);
  output_facet(7,target.medium);
  output_facet(9,target.system_type);
  output_facet(10,target.funct_area);
  output_facet(12,target.setting);
  output_facet(13,target.language);
  output_facet(15,target.other);

}  /* display component
```

```
*****************************************************************************/

/*****************************************************************************
*****************************  output facet  *********************************
*****************************************************************************
*
* This function prints the attribute.
*
*****************************************************************************/
void output_facet(int row, char *str)
{
  gotoxy(row,1);
  sprintf(otext,"%s",str);
  _ot(otext);

}  /* output facet

*****************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

#include "ru.h"
#include <conio.h>

#define Maxbuf            15

typedef struct tbuf {
        char data[FW];
        } TBUFFER;
                                  /* FUNCTION PROTOTYPES */
char *remove_trailing_blanks(char *str);
char *display_choices(char lineno[],char file[]);
char *print_keys(TBUFFER *buffer, int count);

int print_choices(char buffer[][FW], int num);

INVERTNDX find_key(char vdxfile[],char key[]);

void check_thesaurus(char dtafile[],char wrdfile[],char invfile[],
                                                    char vdxfile[]);
void create_filenames(char *dtafile,char *wrdfile,char *invfile,
                                                   char *vdxfile);
void compare_key(INVERTNDX *key,char invfile[]);
void display_keys(INVERTNDX *key,char dtafile[]);
void get_keys(FILE *fp,TBUFFER *buffer);
void invert(char inputfile[],char invfile[],char vdxfile[]);
void thesaurus(char code,char key[],char dtafile[]);

/*************************************************************************
*************************** Thesaurus ***********************************
*************************************************************************
*
* This function is the driver for the common vocabulary of the reuse
* system.
*
*************************************************************************/
void thesaurus(char code,char key[],char dtafile[])
{
  char wrdfile[LEN],invfile[LEN],vdxfile[LEN];
  INVERTNDX ikey;

  create_filenames(dtafile,wrdfile,invfile,vdxfile);

  check_thesaurus(dtafile,wrdfile,invfile,vdxfile);
                                  /* search for key in databank */
  ikey = find_key(vdxfile,key);
```

```
  if (ikey.key[0] == '\0')
    display_keys(&ikey,dtafile);
  else
    compare_key(&ikey,invfile);

  strcpy(key,ikey.key);

}  /* thesaurus

*****************************************************************************/

/*****************************************************************************
************************  check thesaurus  ***********************************
*****************************************************************************
*
* This function insures that the support files for the thesaurus (common
* vocabulary) exist.  If the thesaurus data file does not exist, the
* program will terminate.  If one of the other supporting files does not
* exist, all other files will be created and updated.
*
*****************************************************************************/
void check_thesaurus(char dtafile[],char wrdfile[],char invfile[],
                                                    char vdxfile[])
{
  BOOLEAN flag = TRUE;

  check_path(dtafile,"",1);   /* if dfile !exist, prog will terminate */
  if (check_path(wrdfile,"",2))
    if (check_path(invfile,"",2))
      if (check_path(vdxfile,"",2));
      else flag = FALSE;
    else flag = FALSE;
  else flag = FALSE;

  if (!flag)
  {
    sort(dtafile);
    hyper(dtafile,wrdfile);
    invert(wrdfile,invfile,vdxfile);
  }

}  /* check thesaurus

*****************************************************************************/
```

```
/*************************************************************************
*****************************  compare key  ******************************
*************************************************************************
*
* This function compares the user's input with the primekeys of the
* thesaurus/common vocabulary.  If the user's input != primekey, the
* primekey is substituted in place of the user's input.
*
*************************************************************************/
void compare_key(INVERTNDX *key,char invfile[])
{
  char buffer[Buf_size],
       file[13],
       lineno[LEN] = "",
       primekey[LEN],
       *ptr;
  FILE *fp;
  int count = 0, i;

  open_file(&fp,Read"b",invfile,"compare key");
  fseek(fp,key->offset,SEEK_SET);
  fgets(buffer,Buf_size,fp);
  fclose(fp);

  /* get line #(s) of key occurrences in dta if multiple occurrence, */
  /*  give choices to user */

  ptr = strstr(buffer,":") + 1;   /* discard the key */
  strcpy(buffer,ptr);
  while(buffer[0] != '\0')
  {
    ptr = strstr(buffer,":") + 1;
    strcpy(file,strtok(buffer,": \n\r"));   /* get file name */
    strcpy(buffer,ptr);
    ptr = strstr(buffer,":") + 1;
    strcat(lineno,strtok(buffer,": \n\r")); /* append line# */
    strcat(lineno,",");   /* append delimiter */
    strcpy(buffer,ptr);
  } /* while */

  for (i = 0; i < (int) strlen(lineno); i++)
    if (lineno[i] == ',') count++;

  if (count > 1) strcpy(key->key,display_choices(lineno,file));
  else
  {
    open_file(&fp,Read,file,"compare key");
    count = atoi(strtok(lineno," ,\n\r"));
    for (i = 1; i <= count; i++) fgets(buffer,Buf_size,fp);
    strcpy(key->key,strtok(buffer,",\n\r"));
    if (key->key[0] == '*')        /* wildcard = NULL */
      key->key[0] = '\0';
```

```
    fclose(fp);
  }
}  /* compare key

*****************************************************************************/

/*****************************************************************************
************************  display choices  ***********************************
*****************************************************************************
*
* When a given descriptor is a valid entry for more than one set of
* descriptors in the same thesaurus file, this function displays the
* key words of the common vocabulary when > 1 choices exist for the
* user's input.  The choices are displayed and the user selects the
* closest synonym.  The user's choice is returned to the calling
* function.
*
* EXAMPLE:  list,enumerate,count
*           output,list,write
*   list has two possible meanings so list and output would be presented
*   to the user for his/her selection.
*
*****************************************************************************/
char *display_choices(char lineno[],char file[])
{
  char buffer[Maxbuf][FW],
       temp[Buf_size],
       *ptr;
  FILE *fp;
  int line, num = 0, i;

  open_file(&fp,Read,file,"display choices");
  while(lineno[0] != '\0')        /* extract choices */
  {
    ptr = strstr(lineno,",") + 1;
    strcpy(temp,strtok(lineno,",\n\r"));
    strcpy(lineno,ptr);
    line = atoi(temp);
    rewind(fp);
    for (i = 1; i <= line; i++) fgets(temp,Buf_size,fp);
    sprintf(buffer[num],"%-3c%-16s",++num + 64,strtok(temp,",\n\r"));
  }
  fclose(fp);
                                  /* print the choices */
  do
    line = print_choices(buffer,num);
  while(line == 'n' || line == 'N' || line == 'p' || line == 'P');
  strtok(buffer[line]," ");       /* discard number */
  ptr = strtok(NULL," \n\r");

  if (ptr[0] == '*')               /* wildcard == NULL */
    ptr[0] = '\0';
```

```
  return ptr;

}  /* display choices

*****************************************************************************/

/*****************************************************************************
*************************  print choices  ************************************
*****************************************************************************
*
* This function prints the primekeys that the user has to choose from.
* The user selects and the choice is returned.
*
*****************************************************************************/
int print_choices(char buffer[][FW], int num)
{
  int i, j;
  struct rccoord oldpos;

  windowthes;
  for (i = 1, j = 1; i <= num; i++)
  {
    gotoxy(i,j);
    sprintf(otext,"%-19s",buffer[i]);
    _ot(otext);
  }

  gotoxy(20,1);
  j = getche();

  if (j != 'n' && j != 'N' && j != 'p' && j != 'P')
  {
    j = toupper(j);
    j -= 64;

    while(j < 1 || j > num)
    {
      gotoxy(20,1);
      fflush(stdin);
      j = getche();
      if (j == 'n' || j == 'N' || j == 'p' || j == 'P') break;
      else
      {
        j = toupper(j);
        j -= 64;
      }

    } /* while */
  } /* if j ! */
  clw;
  windowthes;
  clw;
```

```
  return j;

}  /* print choices

*****************************************************************************/

/*****************************************************************************
*************************  display keys  *************************************
*****************************************************************************
*
* This function displays the primekeys from the common vocabulary when
* the user has made an erroneous entry.  The keys are displayed on the
* right side of the screen in groups of 14.  By pressing n or p the user
* can traverse the next or the previous list entries.
*
*****************************************************************************/
void display_keys(INVERTNDX *key,char dtafile[])
{
  char line[Buf_size];
  FILE *fp;
  long linect = 0,                /* line count for data file */
       index;
  TBUFFER *buffer;

  window2;
  _ot("Choose from the given list.\n");
  _ot("Press N for the next list or P for the previous list.  ");

  open_file(&fp,Read,dtafile,"display keys");

  while (fgets(line,Buf_size,fp) != NULL)
    linect++;                     /* count thesaurus entries */
  rewind(fp);

  buffer = (TBUFFER *) calloc((int) linect + 1,sizeof(TBUFFER));

  get_keys(fp,buffer);
  strcpy(key->key,print_keys(buffer,(int) linect));

  if (key->key[0] == '*')  /* wildcard == NULL */
    key->key[0] = '\0';

  fclose(fp);
  window2;
  clw;

}  /* display keys

*****************************************************************************/
```

```
/*************************************************************************
*************************  get keys  *************************************
*************************************************************************
*
* This function reads the prime keys from the vocabulary data file and
* loads them into the buffer.
*
*************************************************************************/
void get_keys(FILE *fp,TBUFFER *buffer)
{
  char line[Buf_size];
  int i = 1;

  while(fgets(line,Buf_size,fp) != NULL)
    strcpy(buffer[i++].data,strtok(line,",\r\n"));

}  /* get keys

*************************************************************************/

/*************************************************************************
*************************  print_keys  ***********************************
*************************************************************************
*
* This function takes the primekeys of the common vocabulary and
* displays them.  The user can traverse the list and make a selection.
*
*************************************************************************/
char *print_keys(TBUFFER *buffer, int count)
{
  char  sbuffer[Maxbuf][FW],
        *dummy;
  int i, j, k, m, max = Maxbuf - 3;

  if (max < count) j = max; else j = count;
  for (i = 1; i <= j; i++)
    sprintf(sbuffer[i],"%-3c%-16s",i + 64,buffer[i].data);

  k = print_choices(sbuffer,i - 1);
  while (TRUE)
  {
    switch(k)
    {
      case 'N':
      case 'n':
        if (i + max > count)
        {
          j = count;
          i = count - max;
          if (i < 1) i = 1;
        }
        else j = i + max;
```

```
        break;

      case 'P':
      case 'p':
        if (i - 2 * max <= 1)
        {
          i = 1;
          if (count < max) j = count;
          else j = max;
        }
        else
        {
          j = i - max;
          i -= 2 * max;
        }
        break;

      default:
        dummy = sbuffer[k] + 3;
        return remove_trailing_blanks(dummy);
        break;
    } /* switch */

    for (m = 1; i <= j; m++, i++)
      sprintf(sbuffer[m],"%-3c%-16s",m + 64,buffer[i].data);

    k = print_choices(sbuffer,m - 1);

  } /* while */
}  /* print keys

*****************************************************************************/

/*****************************************************************************
**********************  create filenames  ***********************************
*****************************************************************************
*
* This function creates the filenames for the thesaurus supporting
* files.
*
* wrdfile is the file containing the locations for each word in
* the thesaurus.
*
* invfile is the inverted list.
*
* vdxfile is the index for the inverted list.
*
*****************************************************************************/
void create_filenames(char *dtafile,char *wrdfile,char *invfile,
                                                  char *vdxfile)
{
  char drive[3], dir[LEN], fname[9], ext[5];
```

```
  _splitpath(dtafile,drive,dir,fname,ext);
  strcpy(wrdfile,fname);
  strcpy(invfile,fname);
  strcpy(vdxfile,fname);
  strcat(wrdfile,".wrd");
  strcat(invfile,".inv");
  strcat(vdxfile,".vdx");
}  /* create filenames

*****************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

#include "ru.h"

void draw_hline(int row, int col, int length);
void draw_vline(int row, int col, int length);
void screens(int screen);

/*************************************************************************
*****************************  screens  ********************************
*************************************************************************
*
* This function is used to subdivide the screen into different text
* windows.  All characters used are in the extended ASCII character set.
*
*************************************************************************/
void screens(int screen)
{
  int i;

  switch (screen)
  {
    case 1:
      window0;
      draw_hline(5,1,80);
      break;

    case 2:
      break;

    case 3:
      break;

    case 4:
      break;

    case 5:
      break;

    case 6:
      break;

    case 7:                          /* thesaurus window */
      window0;
      draw_vline(6,60,16);
      break;
    case 8:                          /* msg window */
```

```
      window0;
      draw_hline(22,1,80);
      break;

    case 9:                          /* openning menu box */
      window0;
      cls;
      sprintf(otext,"Z%.*s?",59,Hline);
      gotoxy(5,10);
      _ot(otext);
      draw_vline(6,10,14);
      draw_vline(6,70,14);
      gotoxy(20,10);
      sprintf(otext,"@%.*sY",59,Hline);
      _ot(otext);
      break;

    default:
      break;
  } /* switch */

}  /* screens

*****************************************************************************/

/*****************************************************************************
************************  draw horizontal line  ************************
*****************************************************************************
*
* This function draws a horizontal line based on the given parameters.
*
*****************************************************************************/
void draw_hline(int row, int col, int length)
{
  sprintf(otext,"%.*s",length,Hline);
  gotoxy(row,col);
  _ot(otext);
}  /* draw horizontal line

*****************************************************************************/
```

```
/*************************************************************************
************************  draw vertical line  ***************************
*************************************************************************
*
* This function draws a vertical line based on the given parameters.
*
*************************************************************************/
void draw_vline(int row, int col, int length)
{
  int i;

  for (i = 0; i < length; i++)
  {
    gotoxy(row + i,col);
    _ot("3");
  }

}  /* draw vertical line

*************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* revised Summer 1991
*
* This program creates the index entries for a hypertext system.  It was
* developed on a 12 Mhz 80286 IBM AT compatible using Microsoft
* Quick C 2.0.  The program is designed to make use of a hard disk.
* As the DOS sort utility is limited to < 64K, the data is read in
* blocks of 16K; this was the block size that produced the quickest run
* time on the development system.  The blocks are sorted and then merged
* in the sort() function.
*
* The hyper() is a road map for the control of this program.  Each
* function is passed the name of the file that is to be manipulated.
* All i/o calls are made from within the respective functions.
*/

#include "ru.h"
#include <malloc.h>
#include <ctype.h>

#define Max_line       500
#define Scr_size       1
#define Word_length  500

                                  /* FUNCTION PROTOTYPES */
void get_words(char input[],char output[]);
void hyper(char file[],char outfile[]);
void unique(char file_name[]);

/*M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M */

void hyper(char file[],char outfile[])
{
                                     /* begin file processing */
  get_words(file,outfile);
  sort(outfile);
  unique(outfile);                /* remove duplicate entries */

} /* hyper */

/*m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m */
```

```
/*************************************************************************
*****************************  get words  ******************************
*************************************************************************
*
* This function takes an input file and breaks it into tokens.  It then
* outputs the tokens along with the file from which it came and the
* appropriate screen number.
*
*************************************************************************/
void get_words(char input[], char output[])
{
  int i;            /* loop control variable */
  int line_number = 0,       /* line number of input file */
      scr_num = 0;           /* screen number */
  char word[Word_length],    /* input word */
       input_line[Max_line],
       *temp_word1,
       seps[] = ",\n\r";          /* separators for strtok() */
  FILE *Finput, *Foutput;

  open_file(&Finput,"r",input,"get_words");
  open_file(&Foutput,"w+",output,"get_words");
                             /* get tokens and arrange output */
  while (fgets(input_line,Max_line-1,Finput) != NULL)
  {                                          /* increment scr_num if necessary */
    if (!(line_number++ % Scr_size)) ++scr_num;

    temp_word1 = strtok(input_line,seps);
    while (temp_word1 != NULL && temp_word1[0] != ' ')
    {
                             /* insure all characters are lower case */
      for (i = 0; i < (int) strlen(temp_word1); i++)
        temp_word1[i] = (char) tolower(temp_word1[i]);

      fprintf(Foutput,"%s : ", temp_word1);
      fprintf(Foutput,"%s : ", input);
      fprintf(Foutput,"%d\n", scr_num);

      temp_word1 = strtok(NULL,seps);  /* get next token */

    } /* while temp word */

  } /* while fgets */

  fclose(Finput);
  fclose(Foutput);

}  /* get words */

/*************************************************************************/
```

```
/*****************************************************************************
*****************************  unique  ***************************************
*****************************************************************************
*
* This function removes all duplicate entries from the given input file.
*
*****************************************************************************/
void unique(char file_name[])
{
  char temp1[Buf_size], temp2[Buf_size];
  FILE *Fwork, *Finput;
                              /* open working file */
  open_file(&Fwork,"w+","work","unique");
  open_file(&Finput,"r",file_name,"unique");

  fgets(temp1,Buf_size,Finput);
  while(!feof(Finput))
  {
    fgets(temp2,Buf_size,Finput);
    fprintf(Fwork,"%s",temp1);
    while((strcmp(temp1,temp2) == 0) && !feof(Finput))
      fgets(temp2,Buf_size,Finput);
    strcpy(temp1,temp2);
  } /* while */

  fclose(Fwork);
  fclose(Finput);
  unlink(file_name);           /* delete the file */
  rename("work",file_name);

}  /* unique */

/*****************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* revised Summer 1991
*
* This program takes a sorted list of words and screen/line references
* and creates an inverted list and index.
*/

#include "ru.h"
                                    /* FUNCTION PROTOTYPES */
char *remove_trailing_blanks(char *str);

void invert(char inputfile[],char invfile[],char vdxfile[]);
void make_inverted_list(char [],char [],char []);

/*M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M */

void invert(char inputfile[],char invfile[],char vdxfile[])
{
  make_inverted_list(inputfile,invfile,vdxfile);
} /* INVERT */

/*m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m */

/*************************************************************************
************************  make inverted list  ************************
*************************************************************************
*
* This function takes the input file and makes the inverted list with
* index.
*
*************************************************************************/
void make_inverted_list(char filename[],char invfile[],char vdxfile[])
{
  char line[LEN],
       input[Buf_size],          /* input word */
       outline[Buf_size],        /* output line of inverted list */
       *word,                      /* following words */
       *remains;                 /* the rest of the line */
  FILE *ifp, *ofp1, *ofp2;
  int count=0;                 /* number of items in index */
  INVERTNDX index;
                               /* open input and output files */
  open_file(&ifp,"r",filename,"mk_inv_lst");
  open_file(&ofp1,"wb+",invfile,"mk_inv_lst");
  open_file(&ofp2,"wb+",vdxfile,"mk_inv_lst");
                    /* get first occurrence of word and check for more */
  if (fgets(line,LEN,ifp) == NULL)
    {
      fcloseall();             /* file is empty */
```

```
      return;
    }
  strcpy(line,strtok(line,"\n\r"));     /* remove cr/lf */
  strcpy(outline,line);
  strcpy(index.key,remove_trailing_blanks(strtok(line,":")));

  while (!feof(ifp))
  {
    if (fgets(line,LEN,ifp) == NULL)
      break;
    word = remove_trailing_blanks(strtok(line,":"));

    while (strcmp(index.key,word) == 0)
    {
      remains = strtok(NULL,"\n\r");   /* remove newline character */
      strcat(outline," :");
      strcat(outline,remains);
      if (fgets(line,LEN,ifp) == NULL)   /* get next word */
        break;
      word = remove_trailing_blanks(strtok(line,":"));
    } /* while strcmp */

    index.offset = ftell(ofp1);
    fprintf(ofp1,"%s\n",outline);
    ++count;
    fwrite(&index,1,sizeof(INVERTNDX),ofp2);
    strcpy(index.key,word);
    remains = strtok(NULL,"\n\r");
    strcpy(outline,index.key);
    strcat(outline," :");
    strcat(outline,remains);

  }  /* while feof */

  index.offset = ftell(ofp1);     /* write last entry to disk */
  fprintf(ofp1,"%s\n",outline);
  ++count;
  fwrite(&index,1,sizeof(INVERTNDX),ofp2);

  fcloseall();

}  /* make inverted list */

/*************************************************************************/
```

```
/*************************************************************************
*******************  remove trailing blanks  *****************************
*************************************************************************
*
* This function takes the input string and removes any trailing blanks.
* The modified string is returned.
*
*************************************************************************/
char *remove_trailing_blanks(char *str)
{

  while (str[strlen(str) - 1] == ' ')
    str[strlen(str) - 1] = '\0';

  return str;

}  /* remove trailing blanks

*************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* revised Summer 1991
*
* This program was developed on a 12 Mhz 80286 IBM AT compatible using
* Microsoft Quick C 2.5.  The program is designed to make use of a hard
* disk.  As the DOS sort utility is limited to < 64K, the data is read
* in blocks of 16K; this was the block size that produced the quickest
* run time on the development system.  The blocks are sorted and then
* merged in the sort() function.
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define buf_size       16000
#define Max_line         500
#define _ot            printf
                              /* FUNCTION PROTOTYPES */
char *merge_input(FILE *input,char *previous,char last_word[],
                                                        int *flag);
void checkpath(char input[], char varname[]);
void merge(char file1[],char file2[],char file3[]);
void sort(char file_name[]);
void openfile(FILE **fp,char type[],char name[],char module[]);

/*************************************************************************
*********************************  sort  *********************************
*************************************************************************
*
* This function makes the necessary calls to the DOS sort utility.
* Data is read in 16k blocks, sorted and then merged.  As the DOS sort
* utility does not provide the "-u" option, a unique() function is
* required if duplicates are to be removed.  The unique function
* should be called immediately following sort.
*
*************************************************************************/
void sort(char file_name[])
{
  unsigned int  x;           /* bytes read into buffer */
  FILE *Fworkin, *Fsorted, *Ftemp, *Ftemp1;
  static char file_buffer[buf_size];
  int y = 0;

  checkpath("sort.exe","PATH");

  openfile(&Ftemp1,"r",file_name,"sort");

  while ((x=fread(file_buffer, 1, buf_size-Max_line, Ftemp1)) > 0)
```

```
  {
    if (!feof(Ftemp1))
    {
      fgets(&file_buffer[x],Max_line-1,Ftemp1);  /* complete the last
                                                  * line of buffer */
      x = x + strlen(&file_buffer[x]);  /* number of bytes read */
    } /* if */

    openfile(&Fworkin,"w","workin","sort");
    fwrite(file_buffer,x,1,Fworkin);
    fclose(Fworkin);
    system("sort < workin > workout");
    merge("workout","sorted","temp");
    unlink("sorted");        /* delete the file */
    rename("temp","sorted");
    unlink("workout");
  } /* while */

  fclose(Ftemp1);
  unlink(file_name);       /* delete the file */
  rename("sorted",file_name);
  unlink("workin");

}  /* sort */

/*************************************************************************/

/*************************************************************************
*****************************  merge  ************************************
*************************************************************************
*
* This function completes a merge of two data sets putting the output
* into the specified file.  This is a cosequential merge that requires
* that both input files be sorted in lexicographical order prior to
* the merge.
*
*************************************************************************/
void merge(char file1[], char file2[], char file3[])
{
  int more_words_exist = 1;
  char line1[Max_line], line2[Max_line], line1p[Max_line] = "",
       line2p[Max_line] = "";
  FILE *Fp1, *Fp2, *Fp3;

  openfile(&Fp1,"r",file1,"merge");
  openfile(&Fp2,"a",file2,"merge");
  rewind(Fp2);
  openfile(&Fp3,"w",file3,"merge");
                            /* initial read */
  strcpy(line1,merge_input(Fp1,line1p,NULL,&more_words_exist));
  strcpy(line2,merge_input(Fp2,line2p,NULL,&more_words_exist));
```

```
  /* while more words exist, if line1 < line2, output line1; else if
   * line1 > line2, output line2 else if == output and get new lines */

  while (more_words_exist)
  {
    if (strcmp(line1,line2) < 0)
    {
      fprintf(Fp3,"%s",line1);
      strcpy(line1,merge_input(Fp1,line1p,line2,&more_words_exist));
    } /* if */
    else if (strcmp(line1,line2) > 0)
         {
           fprintf(Fp3,"%s",line2);
           strcpy(line2,merge_input(Fp2,line2p,line1,&more_words_exist));
         } /* if */
         else
         {
           fprintf(Fp3,"%s",line1);
           strcpy(line1,merge_input(Fp1,line1p,line2,&more_words_exist));
           strcpy(line2,merge_input(Fp2,line2p,line1,&more_words_exist));
         } /* else */

  } /* while */

  fclose(Fp1);
  fclose(Fp2);
  fclose(Fp3);

}  /* merge */

/*************************************************************************/

/*************************************************************************
*************************  merge input  **********************************
*************************************************************************
*
* This function returns the new input called for and the flag that
* determines when both input files are exhausted for functions merge().
* Outputbak keeps a copy of output in its original case form.  This
* allows all comparisons to be based on lower case letters, but it will
* allow the file to maintain case.
*
*************************************************************************/
char *merge_input(FILE *input,char *previous,char last_word[],int *flag)
{
  char high_value[] = "~~~~~~~~~~~~~~~~~~~~",
       output[Max_line] = "",
       outputbak[Max_line],
       *inp,
       temp1[Max_line], temp2[Max_line];

  inp = fgets(output,Max_line,input);
```

```
  if (!inp && (strcmp(last_word,high_value) == 0))
    *flag = 0;            /* both input files empty */
  else
    if (!inp)
      strcpy(output,high_value);   /* input file is empty */
    else
      {
        strcpy(outputbak,output);
        strcpy(temp1,strlwr(output));
        strcpy(temp2,strlwr(previous));

        if (strcmp(temp1,temp2) < 0)
        {
          system("cls");
          _ot("Input file not sorted.  Program terminated.");
          exit(1001);
        }
        strcpy(output,outputbak);
      }

  strcpy(previous,output);
  return output;

}  /* merge input */

/*************************************************************************/

/*************************************************************************
**************************  checkpath  ***********************************
*************************************************************************
*
* Function checkpath insures the input file can be found on the current
* PATH.  If it fails, the program aborts.
*
*************************************************************************/
void checkpath(char input[], char varname[])
{
  char targetfile[13],
       pathname[256] = "";

  strcpy(targetfile,input);
  _searchenv(targetfile,varname,pathname);

  if (pathname[0] == '\0')
  {
    _ot("This program requires the program/file %s to run.\n",input);
    if (strcmp(input,"sort.exe") == 0)
    {
      _ot("Insure the system PATH contains the directory containing");
      _ot(" sort.exe.\n");
      _ot("Check your DOS manual for detailed instructions if ");
      _ot("necessary!\n");
```

```
    }
    else _ot("Insure the file is in the current directory.\n");

    exit(2);

  } /* if pathname */

} /* check path */

/*************************************************************************
**************************  open file  ***********************************
*************************************************************************
*
*  This procedure opens the input and output files for this program.
*
*************************************************************************/
void openfile(FILE **fp,char type[],char name[],char module[])
{
  /*  open input and output files  */
  if ((*fp = fopen(name,type)) == NULL)
  {
    fprintf(stderr,"INPUT/OUTPUT FILE NOT FOUND.\n");
    fprintf(stderr,"File %s not found in function %s.\n",name,module);
    exit(1001);
  }
} /* openfile */

/*************************************************************************/
```

```
/*
* PROGRAMMER:  JOE E. SWANSON
* SSN:  [redacted]
* COMSC 5000  Thesis and Research
* Summer 1991
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define _ot                printf

/*************************************************************************
*****************************  check path  ******************************
*************************************************************************
*
* Function check_path insures the input file can be found on the current
* varname.  If it fails, it returns a flag.
*
* Passed parameter flag = 1, input is required.  Program terminates if
* not found.
* Passed parameter flag = 2, input is optional or existence is being
* confirmed.
* Flag is returned.
*
*************************************************************************/
int check_path(char input[], char varname[], int flag)
{
  char targetfile[13],
       pathname[256] = "";

  strcpy(targetfile,input);
  _searchenv(targetfile,varname,pathname);

  switch(flag)
  {
    case 1:                          /* required files */
      if (pathname[0] == '\0')
      {
        _ot("This program requires the program/file %s to run.\n",
                                                              input);
        if (strcmp(input,"sort.exe") == 0)
        {
          _ot("Insure the system PATH contains the directory \
containing");
          _ot(" sort.exe.\n");
          _ot("Check your DOS manual for detailed instructions if ");
          _ot("necessary!\n");
        }
        else _ot("Insure the file is in the current directory.\n");
        exit(2);
```

```
      } /* if pathname */
      break;

    case 2:            /* optional or check to see if exist */
      if (pathname[0] == '\0') flag = 0;
      else flag = 1;
      break;

    default:  flag = 0;

  } /* switch */

  return flag;

} /* check path */

/*****************************************************************************/
```

APPENDIX D

FUNCTION FACET THESAURUS

```
* NOT APPLICABLE,*
access,accesses
add,increment,total,sum
append,attach,increase
assign,designate
close,release,detach,disconnect,free
compare,test,relate,match,check,verify
complement,negate,invert
compress,shrink,condense,compact
control,controls,command,manipulate,direct,handle,operate,operates
convert,converts
coordinate,coordinates
copy
create
decode
delete
divide
evaluate
exchange,swap
execute,executes
expand
extract,extracts
format
initialize,set up,start
initiate,start
input
insert
join
list,count
load,loads
maintain
measure,advance,size,enumerate,list
modify,change,revise
move,transfer
output,produce
process,processes,filter,prepare
read,reads
save
schedule,schedules
terminate,remove,kill
```

APPENDIX E

OBJECT FACET THESAURUS

```
* NOT APPLICABLE,*
address,addresses
arguments,argument
arrays,array
buffers,buffer
card stack,cards
character,characters,char
descriptors
digits
directories,directory
doubles,double
expressions
files
floats,float
functions
hexadecimal
instruction,command,commands,instructions,inst
integers,integer,int,long int,long,short
interrupts
jobs,job
lines
lists
macros
memory
message
node,nodes
page,pages
processes,process
queue
registers,register
schedules,schedule
simulated disk entry
statistics
string,strings
structure,structures
system
```

APPENDIX F

MEDIUM FACET THESAURUS

```
* NOT APPLICABLE,*
array,arrays
buffer
cards
character,characters,char
disk,disks
double
file
float
integer,long int,long integer,short,long,int,short int
job,jobs
keyboard
line
linked list,deque,deques,list
mouse
node,nodes
operating system,operating systems,os
printer
process cntl block,pcb
process,processes
schedule,schedules
screen
sensor
stack
string,strings
structure,structures
table
tape
tree
```

APPENDIX G

SYSTEM TYPE FACET THESAURUS

```
* NOT APPLICABLE,*
extractor,extractors
generic,general,gen,universal
operating system,operating systems,os
processor,processors
scheduler,schedulers
simulated disk
simulation,simulations
```

APPENDIX H

FUNCTIONAL AREA FACET THESAURUS

```
* NOT APPLICABLE,*
base conversions
context switching
error handling
generic,general,gen,universal
initialization,init
io operations,read,write,output,input
job io,read,write,output,input
job tracing
memory management,mem mgmt
paging
processing
scheduling
simulation
statistics,statistic
string conversion
system synchronization
```

APPENDIX I

SETTING FACET THESAURUS

```
* NOT APPLICABLE,*
academic,scholarly,classical,educational,collegiate,school,edu
generic,general,gen,universal
```

APPENDIX J

LANGUAGE FACET THESAURUS

```
Ada,ada
BASIC,basic
C,c
FORTRAN,fortran
Pascal,pascal
PL/I,pl/1,pl/i
```

APPENDIX K

SAMPLE QUERY RESULTS

```
Sample 1 - six facets
loader

The attributes requested are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Other                 --

*****************************************


The attributes of winldr.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winldr.c matched 5 attributes.
*****************************************


The attributes of loader.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

loader.c matched 5 attributes.
*****************************************
```

```
The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 4 attributes.
****************************************


The attributes of misldr.c are:

        Function              -- load
        Object                -- queue
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misldr.c matched 4 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 3 attributes.
****************************************
```

```
The attributes of misscned.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- acade: '>
        Language              -- C
        Other                 --

missched.c matched 3 attributes.
****************************************


The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 3 attributes.
****************************************


The attributes of j-sched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

j-sched.c matched 3 attributes.
****************************************
```

```
The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 2 attributes.
****************************************


The attributes of winhexc.c are:

        Function              -- convert
        Object                -- digits
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- base conversions
        Setting               -- academic
        Language              -- C
        Other                 --

winhexc.c matched 2 attributes.
****************************************


The attributes of winerror.c are:

        Function              -- output
        Object                -- message
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- error handling
        Setting               -- academic
        Language              -- C
        Other                 --

winerror.c matched 2 attributes.
****************************************
```

```
The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 2 attributes.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 2 attributes.
****************************************


The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 2 attributes.
****************************************
```

```
The attributes of mispgsv.c are:

        Function              -- save
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgsv.c matched 2 attributes.
****************************************


The attributes of mispgflt.c are:

        Function              -- assign
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgflt.c matched 2 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 2 attributes.
****************************************
```

```
The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 2 attributes.
****************************************


The attributes of misinit.c are:

        Function              -- initialize
        Object                -- system
        Medium                -- operating system
        System Type           -- operating system
        Functional Area       -- initialization
        Setting               -- academic
        Language              -- C
        Other                 --

misinit.c matched 2 attributes.
****************************************


The attributes of misfstat.c are:

        Function              -- output
        Object                -- statistics
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- statistics
        Setting               -- academic
        Language              -- C
        Other                 --

misfstat.c matched 2 attributes.
****************************************
```

```
The attributes of disk.c are:

        Function              -- control
        Object                -- simulated disk entry
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- simulation
        Setting               -- academic
        Language              -- C
        Other                 --

disk.c matched 2 attributes.
****************************************


The attributes of cpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

cpu.c matched 2 attributes.
****************************************
```

```
Sample 1a - four facets
loader

The attributes requested are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           --
        Functional Area       -- job io
        Setting               --
        Other                 --

****************************************


The attributes of winldr.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winldr.c matched 3 attributes.
****************************************


The attributes of loader.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

loader.c matched 3 attributes.
****************************************
```

```
The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 2 attributes.
****************************************


The attributes of misldr.c are:

        Function              -- load
        Object                -- queue
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misldr.c matched 2 attributes.
****************************************


The attributes of missched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

missched.c matched 1 attribute.
****************************************
```

```
The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 1 attribute.
****************************************


The attributes of j-sched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

j-sched.c matched 1 attribute.
****************************************
```

```
Sample 2 - six facets
memory

The attributes requested are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Other                 --

****************************************


The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 6 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 6 attributes.
****************************************
```

```
The attributes of memory.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

memory.c matched 6 attributes.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 3 attributes.
****************************************


The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 3 attributes.
****************************************
```

The attributes of misfrpg.c are:

Function	-- close
Object	-- page
Medium	-- array
System Type	-- operating system
Functional Area	-- paging
Setting	-- academic
Language	-- C
Other	--

misfrpg.c matched 3 attributes.
****

The attributes of winhexc.c are:

Function	-- convert
Object	-- digits
Medium	-- integer
System Type	-- operating system
Functional Area	-- base conversions
Setting	-- academic
Language	-- C
Other	--

winhexc.c matched 2 attributes.
****

The attributes of winerror.c are:

Function	-- output
Object	-- message
Medium	-- string
System Type	-- operating system
Functional Area	-- error handling
Setting	-- academic
Language	-- C
Other	--

winerror.c matched 2 attributes.
****

```
The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 2 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 2 attributes.
****************************************


The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 2 attributes.
****************************************
```

```
The attributes of mispgsv.c are:

        Function              -- save
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgsv.c matched 2 attributes.
****************************************


The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 2 attributes.
****************************************


The attributes of mispgflt.c are:

        Function              -- assign
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgflt.c matched 2 attributes.
****************************************
```

```
The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 2 attributes.
****************************************


The attributes of misinit.c are:

        Function              -- initialize
        Object                -- system
        Medium                -- operating system
        System Type           -- operating system
        Functional Area       -- initialization
        Setting               -- academic
        Language              -- C
        Other                 --

misinit.c matched 2 attributes.
****************************************


The attributes of misfstat.c are:

        Function              -- output
        Object                -- statistics
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- statistics
        Setting               -- academic
        Language              -- C
        Other                 --

misfstat.c matched 2 attributes.
****************************************
```

```
The attributes of disk.c are:

        Function              -- control
        Object                -- simulated disk entry
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- simulation
        Setting               -- academic
        Language              -- C
        Other                 --

disk.c matched 2 attributes.
****************************************


The attributes of cpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cnt[illegible]ck
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

cpu.c matched 2 attributes.
****************************************


The attributes of misishex.c are:

        Function              -- compare
        Object                -- hexadecimal
        Medium                -- array
        System Type           -- generic
        Functional Area       -- generic
        Setting               -- generic
        Language              -- C
        Other                 --

misishex.c matched 1 attribute.
****************************************
```

```
Sample 2a - four facets
memory

The attributes requested are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           --
        Functional Area       -- memory management
        Setting               --
        Other                 --

****************************************


The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 4 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 4 attributes.
****************************************
```

```
The attributes of memory.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

memory.c matched 4 attributes.
****************************************


The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 1 attribute.
****************************************


The attributes of misishex.c are:

        Function              -- compare
        Object                -- hexadecimal
        Medium                -- array
        System Type           -- generic
        Functional Area       -- generic
        Setting               -- generic
        Language              -- C
        Other                 --

misishex.c matched 1 attribute.
****************************************
```

The attributes of misfrpg.c are:

Function	-- close
Object	-- page
Medium	-- array
System Type	-- operating system
Functional Area	-- paging
Setting	-- academic
Language	-- C
Other	--

misfrpg.c matched 1 attribute.
****

```
Sample 3 - six facets
cpu

The attributes requested are:

        Function              -- execute
        Object                -- instruction
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Other                 --

****************************************


The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 5 attributes.
****************************************


The attributes of miscpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

miscpu.c matched 5 attributes.
****************************************
```

```
The attributes of cpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

cpu.c matched 5 attributes.
****************************************


The attributes of winterm.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

winterm.c matched 4 attributes.
****************************************


The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 4 attributes.
****************************************
```

```
The attributes of winhexc.c are:

        Function              -- convert
        Object                -- digits
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- base conversions
        Setting               -- academic
        Language              -- C
        Other                 --

winhexc.c matched 3 attributes.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 3 attributes.
****************************************


The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 2 attributes.
****************************************
```

```
The attributes of winerror.c are:

        Function              -- output
        Object                -- message
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- error handling
        Setting               -- academic
        Language              -- C
        Other                 --

winerror.c matched 2 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 2 attributes.
****************************************


The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 2 attributes.
****************************************
```

```
The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 2 attributes.
****************************************


The attributes of mispgsv.c are:

        Function              -- save
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgsv.c matched 2 attributes.
****************************************


The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 2 attributes.
****************************************
```

```
The attributes of mispgflt.c are:

        Function              -- assign
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgflt.c matched 2 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 2 attributes.
****************************************


The attributes of misitoa.c are:

        Function              -- convert
        Object                -- integers
        Medium                -- integer
        System Type           --
        Functional Area       -- string conversion
        Setting               -- academic
        Language              -- C
        Other                 --

misitoa.c matched 2 attributes.
****************************************
```

```
The attributes of misinit.c are:

        Function              -- initialize
        Object                -- system
        Medium                -- operating system
        System Type           -- operating system
        Functional Area       -- initialization
        Setting               -- academic
        Language              -- C
        Other                 --

misinit.c matched 2 attributes.
****************************************


The attributes of misfstat.c are:

        Function              -- output
        Object                -- statistics
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- statistics
        Setting               -- academic
        Language              -- C
        Other                 --

misfstat.c matched 2 attributes.
****************************************


The attributes of disk.c are:

        Function              -- control
        Object                -- simulated disk entry
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- simulation
        Setting               -- academic
        Language              -- C
        Other                 --

disk.c matched 2 attributes.
****************************************
```

The attributes of mismax.c are:

Function	-- compare
Object	-- integers
Medium	-- integer
System Type	-- generic
Functional Area	-- generic
Setting	-- generic
Language	-- C
Other	--

mismax.c matched 1 attribute.
****

```
Sample 3a - four facets
cpu

The attributes requested are:

        Function              -- execute
        Object                -- instruction
        Medium                -- integer
        System Type           --
        Functional Area       -- processing
        Setting               --
        Other                 --

****************************************


The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 3 attributes.
****************************************


The attributes of miscpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

miscpu.c matched 3 attributes.
****************************************
```

```
The attributes of cpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

cpu.c matched 3 attributes.
****************************************


The attributes of winterm.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

winterm.c matched 2 attributes.
****************************************


The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 2 attributes.
****************************************
```

```
The attributes of winhexc.c are:

        Function              -- convert
        Object                -- digits
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- base conversions
        Setting               -- academic
        Language              -- C
        Other                 --

winhexc.c matched 1 attribute.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 1 attribute.
****************************************


The attributes of mismax.c are:

        Function              -- compare
        Object                -- integers
        Medium                -- integer
        System Type           -- generic
        Functional Area       -- generic
        Setting               -- generic
        Language              -- C
        Other                 --

mismax.c matched 1 attribute.
****************************************
```

The attributes of misitoa.c are:

Function	-- convert
Object	-- integers
Medium	-- integer
System Type	--
Functional Area	-- string conversion
Setting	-- academic
Language	-- C
Other	--

misitoa.c matched 1 attribute.
****

```
Sample 4 - six facets
spooler

The attributes requested are:

        Function              -- read
        Object                -- card stack
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Other                 --

****************************************


The attributes of winspool.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winspool.c matched 5 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 5 attributes.
****************************************
```

```
The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 3 attributes.
****************************************


The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 2 attributes.
****************************************


The attributes of winhexc.c are:

        Function              -- convert
        Object                -- digits
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- base conversions
        Setting               -- academic
        Language              -- C
        Other                 --

winhexc.c matched 2 attributes.
****************************************
```

```
The attributes of winerror.c are:

        Function              -- output
        Object                -- message
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- error handling
        Setting               -- academic
        Language              -- C
        Other                 --

winerror.c matched 2 attributes.
****************************************


The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 2 attributes.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 2 attributes.
****************************************
```

```
The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 2 attributes.
****************************************


The attributes of mispgsv.c are:

        Function              -- save
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgsv.c matched 2 attributes.
****************************************


The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 2 attributes.
****************************************
```

```
The attributes of mispgflt.c are:

        Function              -- assign
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgflt.c matched 2 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 2 attributes.
****************************************


The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 2 attributes.
****************************************
```

```
The attributes of misinit.c are:

        Function              -- initialize
        Object                -- system
        Medium                -- operating system
        System Type           -- operating system
        Functional Area       -- initialization
        Setting               -- academic
        Language              -- C
        Other                 --

misinit.c matched 2 attributes.
****************************************


The attributes of misfstat.c are:

        Function              -- output
        Object                -- statistics
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- statistics
        Setting               -- academic
        Language              -- C
        Other                 --

misfstat.c matched 2 attributes.
****************************************


The attributes of disk.c are:

        Function              -- control
        Object                -- simulated disk entry
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- simulation
        Setting               -- academic
        Language              -- C
        Other                 --

disk.c matched 2 attributes.
****************************************
```

The attributes of cpu.c are:

Function	-- execute
Object	-- instruction
Medium	-- process cntl block
System Type	-- operating system
Functional Area	-- processing
Setting	-- academic
Language	-- C
Other	--

cpu.c matched 2 attributes.
****

```
Sample 4a - four facets
spooler

The attributes requested are:

        Function              -- read
        Object                -- card stack
        Medium                --
        System Type           --
        Functional Area       -- job io
        Setting               --
        Other                 --

****************************************


The attributes of winspool.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winspool.c matched 3 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 3 attributes.
****************************************
```

```
Sample 5 - six attributes
scheduler

The attributes requested are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Other                 --

****************************************


The attributes of missched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

missched.c matched 5 attributes.
****************************************


The attributes of j-sched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

j-sched.c matched 5 attributes.
****************************************
```

```
The attributes of winldr.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winldr.c matched 3 attributes.
****************************************


The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 3 attributes.
****************************************


The attributes of loader.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

loader.c matched 3 attributes.
****************************************
```

```
The attributes of winmem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winmem.c matched 2 attributes.
****************************************


The attributes of winhexc.c are:

        Function              -- convert
        Object                -- digits
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- base conversions
        Setting               -- academic
        Language              -- C
        Other                 --

winhexc.c matched 2 attributes.
****************************************


The attributes of winerror.c are:

        Function              -- output
        Object                -- message
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- error handling
        Setting               -- academic
        Language              -- C
        Other                 --

winerror.c matched 2 attributes.
****************************************
```

```
The attributes of wincpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

wincpu.c matched 2 attributes.
****************************************


The attributes of winconv.c are:

        Function              -- convert
        Object                -- address
        Medium                -- integer
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

winconv.c matched 2 attributes.
****************************************


The attributes of spooler.c are:

        Function              -- read
        Object                -- card stack
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

spooler.c matched 2 attributes.
****************************************
```

```
The attributes of misregs.c are:

        Function              -- maintain
        Object                -- registers
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- context switching
        Setting               -- academic
        Language              -- C
        Other                 --

misregs.c matched 2 attributes.
****************************************


The attributes of mispgsv.c are:

        Function              -- save
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgsv.c matched 2 attributes.
****************************************


The attributes of mispgld.c are:

        Function              -- load
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgld.c matched 2 attributes.
****************************************
```

```
The attributes of mispgflt.c are:

        Function              -- assign
        Object                -- page
        Medium                -- buffer
        System Type           -- operating system
        Functional Area       -- paging
        Setting               -- academic
        Language              -- C
        Other                 --

mispgflt.c matched 2 attributes.
****************************************


The attributes of mismem.c are:

        Function              -- access
        Object                -- memory
        Medium                -- array
        System Type           -- operating system
        Functional Area       -- memory management
        Setting               -- academic
        Language              -- C
        Other                 --

mismem.c matched 2 attributes.
****************************************


The attributes of misio.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- io operations
        Setting               -- academic
        Language              -- C
        Other                 --

misio.c matched 2 attributes.
****************************************
```

```
The attributes of misinit.c are:

        Function              -- initialize
        Object                -- system
        Medium                -- operating system
        System Type           -- operating system
        Functional Area       -- initialization
        Setting               -- academic
        Language              -- C
        Other                 --

misinit.c matched 2 attributes.
****************************************


The attributes of misfstat.c are:

        Function              -- output
        Object                -- statistics
        Medium                -- string
        System Type           -- operating system
        Functional Area       -- statistics
        Setting               -- academic
        Language              -- C
        Other                 --

misfstat.c matched 2 attributes.
****************************************


The attributes of disk.c are:

        Function              -- control
        Object                -- simulated disk entry
        Medium                -- structure
        System Type           -- operating system
        Functional Area       -- simulation
        Setting               -- academic
        Language              -- C
        Other                 --

disk.c matched 2 attributes.
****************************************
```

```
The attributes of cpu.c are:

        Function              -- execute
        Object                -- instruction
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- processing
        Setting               -- academic
        Language              -- C
        Other                 --

cpu.c matched 2 attributes.
****************************************
```

```
Sample 5a - four attributes
scheduler

The attributes requested are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           --
        Functional Area       -- scheduling
        Setting               --
        Other                 --

****************************************


The attributes of missched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

missched.c matched 3 attributes.
****************************************


The attributes of j-sched.c are:

        Function              -- schedule
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- scheduling
        Setting               -- academic
        Language              -- C
        Other                 --

j-sched.c matched 3 attributes.
****************************************
```

```
The attributes of winldr.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

winldr.c matched 1 attribute.
****************************************


The attributes of misspool.c are:

        Function              -- terminate
        Object                -- jobs
        Medium                -- process cntl block
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

misspool.c matched 1 attribute.
****************************************


The attributes of loader.c are:

        Function              -- load
        Object                -- jobs
        Medium                --
        System Type           -- operating system
        Functional Area       -- job io
        Setting               -- academic
        Language              -- C
        Other                 --

loader.c matched 1 attribute.
****************************************
```

VITA

Joe E. Swanson

Candidate for the Degree of

Master of Science

Thesis: A REUSABLE SOFTWARE CATALOG INTERFACE

Major Field: Computer Science

Biographical:

Personal Data: [redacted]

Civilian Education: Graduated from Guthrie High School, Guthrie, Oklahoma in May 1977; received Bachelor of Music Degree from Central State University at Edmond in May 1981; completed requirements for the Master of Science degree at Oklahoma State University in December 1991.

Military Education: Graduate of the following courses: Transportation Officer Basic Course, August 1981; Initial Entry Rotary Wing Course, May 1982; Aviation Maintenance Officer Course, August 1983; Aviation Officer Advanced Course, March 1986; Combined Arms and Services Staff School, August 1989.

Professional Experience: Currently employed by the United States Army, has served in the following positions: Aviation Maintenance Officer, Ft. Hood, Texas, January 1983 to September 1985. Commander, Aviation Maintenance Company, Ft. Rucker, Alabama, April 1986 to May 1988. Instructor, Aviation Officer Advanced Course, Ft. Rucker, Alabama, May 1988 to May 1989. Will assume duties as Chief, Computer Support Division, Combined Forces Command, Seoul, Korea, September 1991.